OPTIMIZACION DEL COSTO DE ENLACES EN UNA RED

Victor Hugo Sanjinez

Tutor: Mikel Izal Azcarate

Titulación: Ingeniería Informática

Pamplona, 1 de julio de 2011

Agradecimientos

Un especial agradecimiento a mis docentes,
tanto de la Universidad Mayor, Real y Pontificia de San Francisco Xavier
como de la Universidad Pública de Navarra, quienes me
inculcaron sus conocimientos a lo largo de mi carrera académica.

Dedicatoria

A mi madre, por su apoyo incondicional.
A mis amigos y amigas por su ayuda y consejos.

Resumen

El proyecto de fin de carrera (PFC) desarrolla un software para encontrar los enlaces óptimos para interconectar los nodos de una red determinada, aplicando una técnica de Inteligencia Artificial como son los Algoritmos Genéticos los cuales forman parte de la rama del Soft Computing.

El software esta formado por los siguientes módulos:

- Modulo para cargar y leer los datos introducidos por el usuario que describen el problema o su equivalente generador de problemas mediante el uso de datos aleatorios

- Algoritmo Genético que determine las terminales o dispositivos de telecomunicación que conformen el núcleo Backbone de una red. Mediante la formación de grupos de nodos, en los cuales exista simetría en el ancho de banda requerido

- Algoritmo Genético que determine el costo mínimo para interconectar las terminales o dispositivos principales cumpliendo con los requerimientos que cada terminal especifica

- Interfaz grafica para presentar los resultados, en la cual se puedan realizar ajustes a la solución aproximada

- Modulo para generar los reportes correspondientes al cálculo de la solución final

- Modulo de pruebas para comprobar la capacidad del software

Siendo los más importantes, los Algoritmos Genéticos desarrollados para determinar el núcleo Backbone de la red, y los enlaces necesarios para interconectar los nodos con un costo mínimo.

Las pruebas realizadas al funcionamiento del software nos indican el rendimiento y el tiempo requerido para encontrar una solución.

El software es programado en MATLAB, de manera que puede ejecutarse tanto en plataformas Windows como Unix.

La planificación del PFC esta pensado para ser utilizada por diseñadores de redes, analistas de redes, administradores de redes y como un programa ejemplo con fines académicos.

Índice

1. **Introducción y propósito del PFC**
   - 1.1 Antecedentes
   - 1.2 Problema
   - 1.3 Propósito
   - 1.4 Objetivos
     - 1.4.1 Objetivos Generales
     - 1.4.2 Objetivos Específicos
2. **Algoritmos Genéticos**
   - 2.1 Principios
   - 2.2 Concepto de Algoritmo Genético
   - 2.3 Componentes de Algoritmo Genético
     - 2.3.1 Codificación y/o Representación
     - 2.3.2 Genotipo
     - 2.3.3 Fenotipo
     - 2.3.4 Gen
     - 2.3.5 Cromosoma
     - 2.3.6 Población
   - 2.4 Funcionamiento de un Algoritmo Genético
     - 2.4.1 Inicialización
     - 2.4.2 Selección
       - 2.4.2.1 Probabilística
       - 2.4.2.2 Torneo
       - 2.4.2.3 Orden
       - 2.4.2.4 Aleatoria
       - 2.4.2.5 Generacional
       - 2.4.2.6 Por estado estacionario
       - 2.4.2.7 Jerárquica
     - 2.4.3 Reproducción
       - 2.4.3.1 Cruce
         - 2.4.3.1.1 Basado en puntos
         - 2.4.3.1.2 Uniforme
         - 2.4.3.1.3 Scattered
       - 2.4.3.2 Mutación
         - 2.4.3.2.1 Bit
         - 2.4.3.2.2 MultiBit
     - 2.4.4 Evaluación
3. **Desarrollo**
   - 3.1 Modulo para introducir datos
     - 3.1.1 Generar Datos Aleatorios
     - 3.1.2 Cargar Datos de Archivos
   - 3.2 Algoritmo Genético para determinar el núcleo de nodos backbone
     - 3.2.1 Generar datos iniciales
     - 3.2.2 Generar población inicial
     - 3.2.3 Selección de cromosomas
     - 3.2.4 Generar población según criterio

3.2.5 Generar nuevo cromosomas mediante operaciones de cruce y mutación
3.2.6 Evaluación mediante desviación estándar
3.2.7 Gravedad de Reilly
3.3 Algoritmo Genético que determine el costo mínimo para interconectar en núcleo de nodos backbone
3.3.1 Generar población inicial
3.3.2 Selección de cromosomas
3.3.3 Generar población según criterio
3.3.4 Generar nuevo cromosoma mediante operaciones de cruce y mutación
3.3.5 Evaluación según valor asignado
3.3.6 Algoritmo Dijkstra
3.3.7 Redes Homogéneas
3.3.8 M/M/1
3.3.9 Delay Time
3.4 Interfaz Grafica para presentar los resultados
3.5 Modulo para generar reportes
3.5.1 Formato Excel
3.5.2 Formato CSV
4. Gestión del Proyecto
5. Pruebas
6. Conclusiones
7. Bibliografía
8. Referencia Bibliográfica
9. Anexo I

1 Introducción y propósito del PFC

1.1 Antecedentes

Desde las primeras redes de área local (LAN), el diseño de redes ha incrementado su complejidad debido:

- al crecimiento de los requerimientos de ancho de banda por parte de los usuarios
- la magnitud del área geográfica que una red puede abarcar, por ejemplo redes de área metropolitana (MAN), redes de área amplia (WAN)
- seleccionar el tipo de topología adecuada para una red, como puede ser Bus, Anillo, Estrella, Malla
- elegir el tipo de medio para trasportar los datos, ya sea cable de cobre, fibra óptica, conexiones dedicadas. etc.

Se debe tomar las decisiones apropiadas para obtener un alto rendimiento, donde la red pueda ser tolerante a fallas, con calidad de servicio y tenga la capacidad de una futura expansión. Todas estas características que se buscan en una red deben ser alcanzadas con un costo mínimo. [10].

El diseño de redes de comunicación como se puede apreciar es una tarea que implica varios aspectos a tomar en cuenta para poder encontrar una solución a un conjunto de requisitos determinados.

Uno de los aspectos son los enlaces principales que interconectan los nodos, los cuales implican un costo en su instalación, mantenimiento, etc. Determinar los enlaces que existen en una red, se convierte en parte fundamental del diseño de redes de comunicación, tarea relativamente simple, pero a medida que el número de nodos se incrementa la complejidad para diseñar la red va aumentando exponencialmente, llegando un punto en el que el tiempo empleado en el diseño de la red, influye en los tiempos de un proyecto y origina que el proceso sea ineficiente.

Para determinar los enlaces de una red, es posible utilizar diversas técnicas matemáticas, lógica tradicional, experiencia profesional, inteligencia artificial, etc.

La inteligencia artificial al tratar de mejorar el rendimiento de una labor desempeñada por una persona mediante procesos que imiten o simulen el razonamiento humano, el comportamiento de las especies en la naturaleza, etc. Puede convertirse en una técnica robusta, confiable, y optima para encontrar soluciones adecuadas a los problemas que se puedan presentar.

A través de los grupos que la conforman como ser el Soft Computing y la búsqueda de soluciones aproximadas a problemas con información difusa, imprecisa, variante.
Mediante el uso de un grupo de técnicas por ejemplo las redes neuronales, lógica difusa, algoritmos genéticos, etc. [8].

1.2 Problema

Uno de los problemas del diseño de redes, es tratar de determinar el mínimo costo de los enlaces en una red, ya que dicho costo es proporcional a la cantidad de enlaces y su capacidad, sin embargo encontrar la solución computacional a este problema puede convertirse en un proceso que implique un tiempo de espera elevado, es por ello necesario optimizar este proceso para obtener así una solución rápida y eficiente.

Técnicamente el problema radica en la complejidad para determinar los enlaces adecuados para interconectar una red con el mínimo costo posible y la lentitud que este proceso implica.

1.3 Propósito

El propósito del PFC es el estudio e implementación de un software aplicando una técnica de Soft Computing como son los Algoritmos Genéticos, para encontrar una solución más rápida en comparación a la Programación Lineal.

El proyecto mejorara el aprovechamiento de los recursos humanos ya que el software calculara una solución aproximada al problema, reduciendo en gran medida el esfuerzo empleado al analizar o diseñar una red.

El desarrollo del software aplicando Algoritmos Genéticos constituye un aporte académico, porque se convierte en un ejemplo práctico que muestra como emplear la técnica de Soft Computing en un problema real. El aporte toma mayor relevancia si tomamos en cuenta que los Algoritmos Genéticos son una rama de la Inteligencia Artificial ampliamente utilizada y con resultados satisfactorios como lo demuestran por ejemplo las siguientes investigaciones:

Algoritmo genético desarrollado en común por ingenieros de General Electric y el Rensselaer Polytechnic Institute produjo el diseño de la turbina de un motor a reacción de altas prestaciones que era tres veces mejor que la configuración diseñada por humanos, y un 50% mejor que una configuración diseñada por un sistema experto que recorrió con éxito un espacio de soluciones que contenía más de 10.387 posibilidades. [1].

Algoritmos genéticos para diseñar una sala de conciertos con propiedades acústicas óptimas, maximizando la calidad del sonido para la audiencia, para el director y para los músicos del escenario. [2].

Algoritmos evolutivos para evolucionar un complejo sistema de reconocimiento de patrones donde la tarea principal es clasificar aviones basándose en sus reflexiones radar. Un mismo tipo de avión puede devolver señales bastante distintas dependiendo del ángulo y elevación desde el que se le observa, y distintos tipos de avión pueden devolver señales muy parecidas, así que esto no es una tarea trivial. El sistema de reconocimiento de patrones evolucionado clasificó correctamente un 97,2% de los objetivos, un porcentaje neto mayor que el de las otras tres técnicas:

- *una red neuronal perceptrón*
- *un algoritmo clasificador KNN*
- *un algoritmo de base radial*

La precisión de la red de base radial era sólo un 0,5% menor que la del clasificador evolucionado, una diferencia que no es estadísticamente significativa, pero la red de base radial necesitó 256 detectores de característica mientras que el sistema reconocedor evolucionado sólo utilizó 17. [3].

Por los resultados obtenidos en las investigaciones realizadas, los Algoritmos Genéticos son una alternativa viable para tomar en cuenta al momento de elegir una técnica para dar una solución a un problema.

Así mismo el software aplica los Algoritmos Genéticos debido a las grandes ventajas que presenta:

- Son algoritmos adaptativos capaces de solucionar problemas que cambian en el tiempo
- Operan de forma simultánea con varias soluciones en lugar de trabajar de forma secuencial como en las técnicas tradicionales
- Gracias al paralelismo que conllevan los algoritmos genéticos, es posible evitar convergencias hacia falsas soluciones (máximos locales)
- Funcionan bien resolviendo problemas cuyo espacio de soluciones es vasto para hacer una búsqueda exhaustiva en un tiempo razonable
- No requieren tener conocimiento profundo del problema a resolver
- Usa operadores probabilísticos en lugar de operadores determinísticos tradicionales.
- Por la estructura del Algoritmo Genético es posible ejecutarlos en arquitecturas paralelas consiguiendo mayor rendimiento [6][7][9].

El proyecto se desarrolla con el software MATLAB, dadas las ventajas que presenta:

- Software matemático con un entorno de desarrollo integrado IDE
- Funciones para el tratamiento de matrices
- Representación de datos y funciones
- Creación de interfaces de usuario
- Lenguaje propio, lenguaje M
- Multiplataforma, Windows, Unix, etc.

El software permitirá minimizar el costo de los enlaces en una red, por consiguiente se puede tener una idea del impacto económico que la solución aproximada propone. Ya que al garantizar que los requerimientos serán cubiertos empleando los enlaces adecuados, se reduce los gastos de implementación y mantenimiento de la red.

1.4 Objetivos

1.4.1 Objetivos Generales

El objetivo general es desarrollar un software que permita optimizar y obtener de una manera rápida el mínimo costo de los enlaces de una red aplicando algoritmos genéticos.

1.4.2 Objetivos Específicos

- Implementar un modulo para cargar y leer los datos introducidos por el usuario que describen el problema o su equivalente generador de problemas mediante el uso de datos aleatorios

- Implementar un Algoritmo Genético que determine las terminales o dispositivos de telecomunicación que conformen el núcleo Backbone de una red. Mediante la formación de grupos de nodos, en los cuales exista simetría en el ancho de banda requerido

- Implementar un Algoritmo Genético que determine el costo mínimo para interconectar las terminales o dispositivos principales cumpliendo con los requerimientos que cada terminal especifica

- Implementar una interfaz grafica para presentar los resultados, en la cual se puedan realizar ajustes a la solución aproximada

- Desarrollar un modulo para generar los reportes correspondientes al cálculo de la solución final

- Desarrollar un modulo de pruebas para comprobar la capacidad del software

2 Algoritmos Genéticos

2.1 Principios

Basándose en las teorías de Darwin sobre el origen de las especies. Los algoritmos genéticos surgen logrando adaptar los principios teóricos de la evolución biológica a problemas reales de nuestro medio, imitando el proceso de la selección natural como estrategia para encontrar soluciones aproximadas.

Los inicios de los algoritmos genéticos son a principios de la década de los 60 con el desarrollo de algoritmos inspirados en la optimización de funciones y aprendizaje automático. Con el tiempo la estrategia evolutiva adiciono componentes a su esquema hasta convertirse en la programación evolutiva actual, el cual integra el estudio de los fundamentos y aplicaciones de técnicas heurísticas basadas en los principios de la evolución natural.

Dichas técnicas dan lugar a lo que se conoce como ecuación evolutiva.

Computación =	Algoritmos	+ Estrategias	+ Programación	
Evolutiva	Genéticos	Evolutivas	Evolutiva	

Todo lo referente a la Computación evolutiva contiene una base biológica, esta trata de imitar el proceso de selección natural, asimismo la computación evolutiva pertenece al campo de la investigación en Inteligencia Artificial denominado Soft Computing.

Soft	= Computación	+ Redes Neuronales	+ Lógica	
Computing	Evolutiva	Artificiales	Difusa	

Dentro de la informática podemos describir dos tipos de grupos:

- Hard Computing
- Soft Computing

El Hard Computing trata de dar soluciones exactas y precisas, haciendo uso de métodos analíticos y técnicas matemáticas convencionales.

En cambio el Soft Computing busca soluciones aproximadas, mediante el uso de la información incompleta, la incertidumbre y la inexactitud, explotando y explorando el espacio de soluciones hasta encontrar una convergencia aproximada, la cual es la solución que mejor se adapte al medio.

De entre los investigadores que participaron en el marco teórico y la consolidación de los algoritmos genéticos, es John Holland el mayor referente ya que fue el primero en proponer los operadores de reproducción trabajo que realizo mientras pertenecía a la Universidad de Michigan, los cuales permiten convertir un conjunto de conceptos en un esquema ideal para encontrar soluciones. [8].

2.2 Concepto de Algoritmo Genético

Un algoritmo genético es un método adaptativo, que encuentra soluciones en un espacio extenso de posibles soluciones, generalmente usado en problemas de búsqueda y optimización de parámetros.

Este tipo de algoritmos están inspirados en la evolución biológica, un proceso de evolución basado en el principio de la selección natural y la reproducción de Darwin, en el cual se mantiene la supervivencia del individuo que mejor se adapte al medio.

2.3 Componentes de Algoritmo Genético

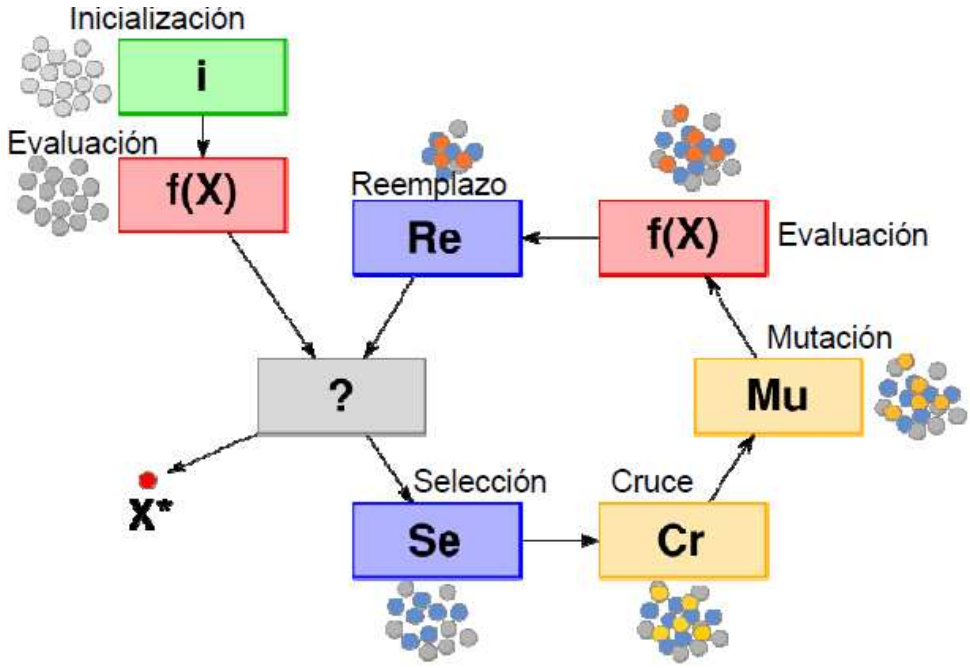


Figura 1 – Estructura de un Algoritmo Genético [18].

Un algoritmo genético presenta una estructura definida como se muestra en la Figura 1, la cual proporciona las funciones necesarias para encontrar una solución a un problema específico.
La estructura compuesta por un conjunto de componentes, en los cuales cada uno de ellos tiene una función especial, e integrados entre todos, llegan a conformar una estructura apta, la cual es el núcleo del algoritmo genético.
Las funciones que los componentes desempeñan, pueden ir desde seleccionar que individuos pasaran a la próxima generación hasta realizar las operaciones de reproducción para crear nuevos individuos.

2.3.1 Codificación y/o Representación

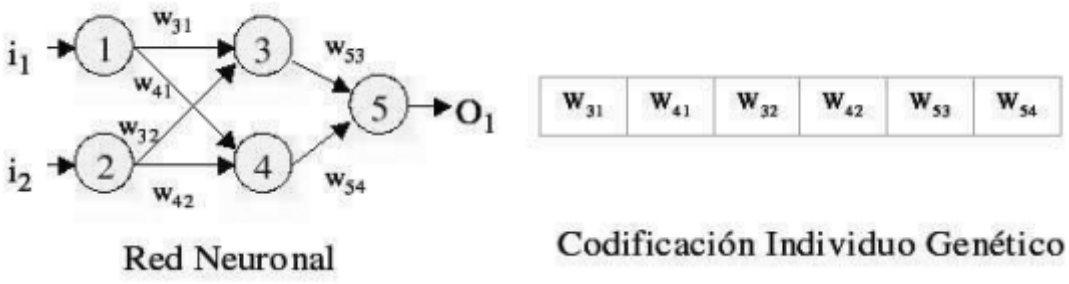


Figura 2 – Codificación de una Red Neuronal [8].

Las soluciones potenciales a un problema pueden ser presentadas como un conjunto de parámetros.
En la Figura 2, se muestra como los parámetros pueden estar formados por valores binarios, enteros, reales, incluso pueden existir codificaciones las cuales no estén representadas por valores simples, sino por estructura de datos, como es el caso de la programación evolutiva.
La codificación tiene una elevada importancia ya que define los operadores con los que trabaja el algoritmo genético, es decir el tipo de selección, reproducción y evaluación que es empleado con los individuos.

2.3.2 Genotipo

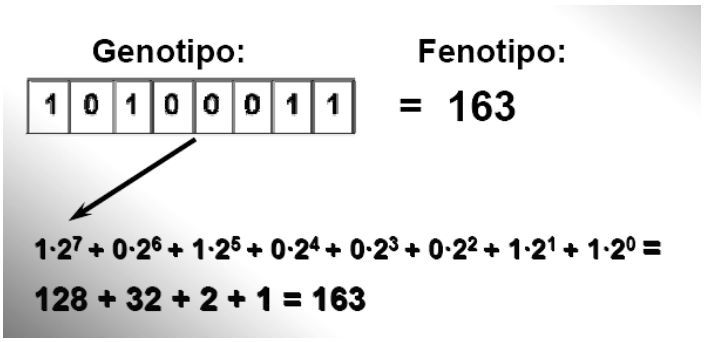


Figura 3 – Concepto de Genotipo y Fenotipo [19].

El genotipo es un conjunto de parámetros que representan una solución válida para un problema específico.
Es decir, cuando se necesita representar una solución del mundo real al informático, se hace mediante parámetros (números, letras, estructuras, etc.).
El proceso de llevar lo real al mundo informático, a través de un conjunto de parámetros es denominado genotipo, porque es el genotipo el que representa algo real, significativo, etc.

Como muestra la Figura 3, el genotipo contiene la información genética necesaria para construir el organismo, es decir para construir la solución real.

2.3.3 Fenotipo

En oposición al genotipo, el fenotipo es lo real, es el individuo, es el ente final significativo que está representado a través de un grupo de parámetros por el genotipo.

Por ejemplo, en términos biológicos, la información genética contenida en el ADN de un individuo correspondería al genotipo, mientras que la expresión de ese ADN es decir el propio individuo vendría a ser el fenotipo.

2.3.4 Gen

Es la representación mínima en una solución, es decir es un parámetro o característica que acoplado a un conjunto de genes forman la solución válida a un problema especifico.

2.3.5 Cromosoma

Es la representación de una solución, conformado por un conjunto de genes, se convierte en una cadena de valores o parámetros la cual contiene la información necesaria para representar una solución real al problema.

2.3.6 Población

Es un conjunto de soluciones es decir cromosomas, el cual con el paso de las iteraciones, presenta generaciones cada vez mejor adaptadas al medio.
El número de individuos por población puede influir en un algoritmo genético, ya que si la cantidad es pequeña existe el riesgo de representar completamente al problema, en cambio si la cantidad es elevada, provoca que el coste computacional necesario para encontrar una solución se incremente.

En general, el tamaño de la población está comprendido entre L y 2L, donde L toma como referencia la longitud de un cromosoma.

2.4 Funcionamiento de un Algoritmo Genético

Los algoritmos genéticos trabajan sobre una población de individuos, cada uno de ellos representa una posible solución a un problema que se desea resolver.

Un individuo tiene relacionado un valor el cual representa que tan adaptado esta al medio.

Las generaciones formadas inicialmente a partir de una población generada con parámetros aleatorios, con el pasar de las iteraciones y aplicando los operadores sobre los individuos que la conforman, presentaran individuos mejor adaptados al medio que los pertenecientes a las poblaciones anteriores.

Los operadores que nos proporcionan el poder generar nuevos individuos, son:

- Selección de un conjunto de individuos
- Reproducción de los individuos seleccionados
- Evaluación de los nuevos individuos generados

El funcionamiento genérico de un algoritmo genético puede apreciarse en la siguiente figura:

```
Inicializar población actual aleatoriamente

MIENTRAS no se cumpla el criterio de terminación
      crear población temporal vacía

  MIENTRAS  población temporal no llena
        seleccionar padres
        cruzar padres con probabilidad Pc
        SI se ha producido el cruce
            mutar uno de los descendientes con probabilidad Pm
            evaluar descendientes
            añadir descendientes a la población temporal
        SINO
            añadir padres a la población temporal
        FIN SI
  FIN MIENTRAS

  aumentar contador generaciones
  establecer como nueva población actual la población temporal

FIN MIENTRAS
```

Figura 4 – Estructura de un Algoritmo Genético [8].

2.4.1 Inicialización

Consiste en generar una población inicial con cromosomas formados por genes con valores aleatorios.
Dicha población es la raíz de las futuras soluciones que se generaran con la aplicación de los operadores genéticos, la población evolucionara tomando como base los esquemas propuestos por Darwin sobre la selección natural y los individuos se adaptaran en mayor medida tras el paso de cada generación.

2.4.2 Selección

Los algoritmos de selección son los encargados de elegir los individuos que podrán reproducirse.
Al tratar los algoritmos genéticos de imitar el proceso que ocurre en la naturaleza, se dará mayor posibilidades de reproducción a los individuos que estén mejor adaptados al medio, es decir, aquellos cromosomas que después de la evaluación cuenten con un valor de adaptación más aproximado al buscado, dichos cromosomas serán los que tengan mayor probabilidad de pasar a la siguiente generación al transmitir su información genética a los hijos que puedan procrear.

Muchos eventos pueden suceder dependiendo del criterio de selección que se emplee, desde una convergencia rápida a una solución con probabilidad similar de ser la correcta o no por solo seleccionar a los mejores individuos, hasta una iteración infinita por no conseguir una convergencia producto de una selección aleatoria de individuos.

Es por ello, que es muy importante emplear un método de selección adecuado para el problema que se desea resolver.

Dentro de los métodos de selección pueden describirse dos grupos:

- Métodos probabilísticos
- Métodos determinísticos

Los métodos probabilísticos asocian un número de posibilidades de reproducción a cada individuo, utilizando para la selección de los mismos un valor aleatorio, el cual servirá para identificar los individuos que fueron seleccionados.

Los métodos determinísticos también asignan un valor a cada individuo, sin embargo dicho valor será el número de veces que podrá reproducirse, controlando el número de veces que un individuo se reproduce, se puede evitar la predominancia de algunos individuos mejor adaptados al medio, pero también se excluye la posibilidad de que cada individuo pueda reproducirse de acuerdo a la evolución que este mismo tenga.

A continuación se procede a describir con mayor detalle los métodos de selección mas utilizados:

2.4.2.1 Probabilística

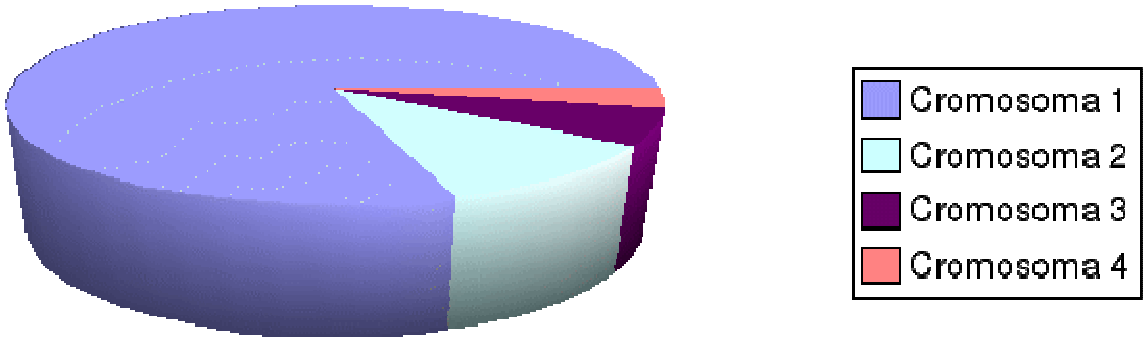


Figura 5 – Asignación de probabilidades

A cada uno de los individuos de la población se le asigna una parte proporcional según su valor de aptitud, de tal forma que la suma de todas las partes sea la unidad, de esta manera los mejores individuos recibirán una mayor probabilidad de ser seleccionados, todo lo contrario a los individuos con un valor de aptitud alejado de la solución, dicho concepto está representado en la Figura 5.

Para seleccionar un individuo se genera un numero aleatorio entre los intervalos [0,1], y se devuelve el individuo situado en el rango correspondiente en la asignación de probabilidades.

El método es muy sencillo y posiblemente el más usado, sin embargo tiene el inconveniente de provocar una rápida convergencia, ya que los mejores individuos para la

generación actual, acaparan las posibilidades de reproducción, relegando a los individuos menos adaptados al medio, esto reduce en gran medida el espacio de soluciones ya que se evita el ingreso de nueva información genética a la generación futura.

2.4.2.2 Torneo

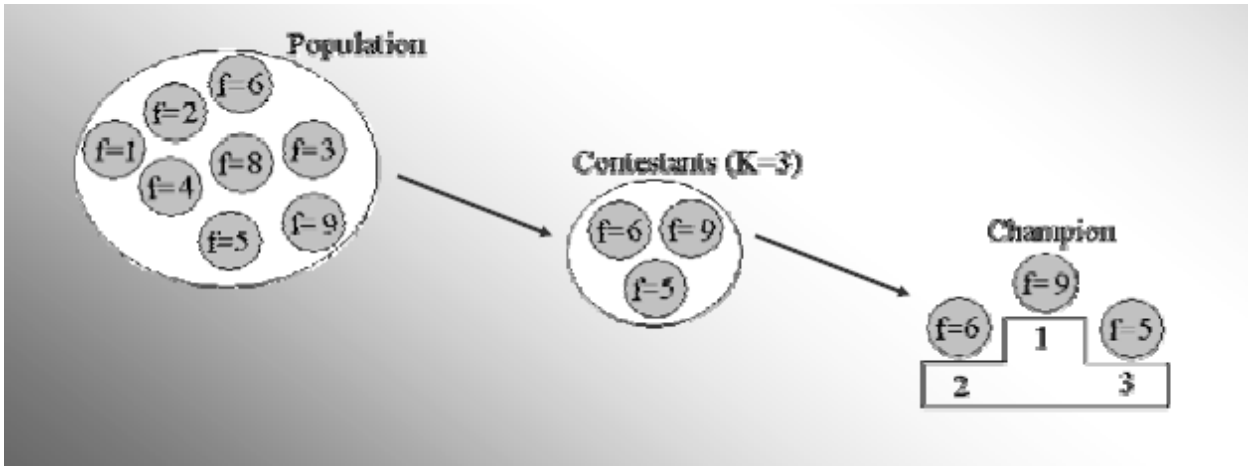


Figura 6 – Proceso de selección método Torneo [19].

Se selecciona un número determinado de individuos, de entre los cuales se elige al más apto para pasar a la siguiente generación, el proceso se muestra en la Figura 6.

El método proporciona la posibilidad de una selección aun más proporcional, porque se da la posibilidad a los individuos menos adaptados de ser seleccionados.

Dependiendo del número de individuos por grupo, se puede modificar la presión de selección, es decir, cuando participan muchos individuos en cada torneo, la presión de selección es elevada y los peores individuos tienen pocas oportunidades de reproducción, y cuando el tamaño del torneo es reducido, la presión de selección disminuye y los peores individuos tienen más oportunidades de ser seleccionados.

Si se opta por un método con una alta presión de selección se centra la búsqueda de las soluciones en un entorno próximo a las mejores soluciones actuales, por el contrario si se opta por una presión de selección menor se deja el camino abierto para la exploración de nuevas regiones del espacio de posibles soluciones.

2.4.2.3 Orden

Los individuos se ordenan según su valor de adaptación, de mejor a peor, de esta manera el lugar que ocupa en la lista se denomina el orden del cromosoma.

Con este método se evita la aparición de individuos, que aceleren la convergencia hacia una respuesta.

Para seleccionar un individuo solo basta generar un numero ramdomico entre los intervalos del orden correspondiente, y devolver el individuo situado en el orden del cromosoma.

2.4.2.4 Aleatoria

En el método solo interviene el azar, sin tomar en cuenta valores de aptitud, ni parámetros adicionales, se selecciona los individuos de acuerdo a un valor aleatorio.

La bondad de este método es el poder dar la oportunidad a todos y cada uno de los individuos de ser seleccionando, lo cual implica explorar el espacio de soluciones.

Sin embargo al ser totalmente aleatorio, puede influir negativamente en la convergencia hacia la solución esperada.

2.4.2.5 Generacional

La descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación, es decir no se conservan los individuos entre las generaciones.

Como consecuencia se pierda el trabajo realizado anteriormente, ya que en cada generación se comenzara a explorar el espacio de soluciones con una nueva población, dicho proceso no garantía de que la siguiente generación sea mejor que la actual, por consiguiente el método puede ser perjudicial.

2.4.2.6 Por estado estacionario

Los individuos seleccionados se mantienen en la población existente, reemplazando a algunos de los miembros menos aptos de la siguiente generación, de esta manera se conservan algunos individuos entre generaciones.

El hecho de conservar a los mejores individuos entre generaciones, nos garantiza que los resultados de las siguientes generaciones serán iguales o mejores que la actual, ya que en ningún caso se podrá regresar a una soluciones menos adaptada que la actual.

2.4.2.7 Jerárquica

Los individuos atraviesan múltiples rondas de selección en cada generación, las evaluaciones de los niveles más bajos, son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente.

Aplicando este método se reduce el tiempo de procesamiento, ya que para evaluar a los individuos menos adaptados la evaluación es más rápida, mientras que para los individuos que tienen mayores aptitudes la evaluación es más rigurosa.

2.4.3 Reproducción

Una vez seleccionados los individuos, los mismos son recombinados para producir la descendencia que se insertara en la siguiente generación, dicha generación llevara en su interior, es decir en cada uno de los cromosomas, la mezcla de la información genética de los individuos de la población precederá.

Para producir este cruce de información genética necesitamos métodos de reproducción los cuales caracterizan el proceso de selección natural y la evolución de Darwin.
Dentro de la reproducción existen dos conceptos, la explotación y la exploración, el conseguir el mayor beneficio de un individuo es asociado a la explotación por medio de operadores de cruce.
Ahora bien, introducir nueva información genética en un individuo corresponde a la exploración, la cual es conseguida con operadores de mutación.

2.4.3.1 Cruce

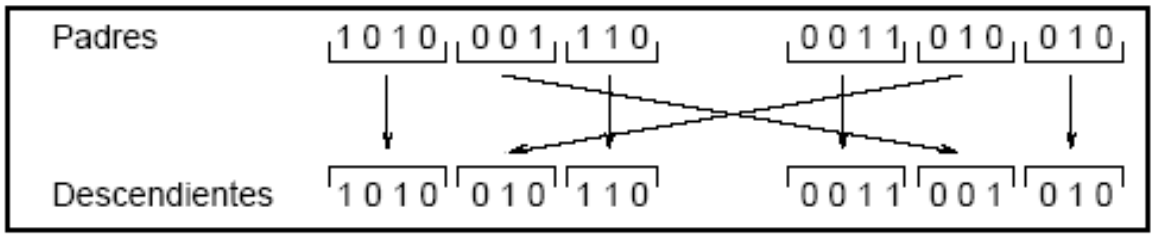


Figura 7 – Ejemplo de cruce uniforme [19].

El método de cruce, es la forma de calcular el genoma del nuevo individuo en función del genoma de los padres, de esta manera los hijos heredan características de los padres, como muestra la Figura 7.

La importancia del método de cruce, para la transición entre generaciones es elevada ya que la probabilidad de cruce con la que se trabajo ronda el 90 porciento.

La idea principal del cruce se basa en que si se toman dos individuos correctamente adaptados al medio y se obtiene una descendencia que comparta genes de ambos individuos, existe la posibilidad de que los genes heredados conformen nuevos individuos mejor adaptados al medio.

2.4.3.1.1 Basado en puntos

De los métodos de cruce, es el más sencillo, consiste en seleccionar aleatoriamente uno o varios puntos, de los cromosomas del padre, para generar segmentos en ambos padres los cuales serán intercambiados entre los hijos, de esta manera los descendientes heredan la información genética de los padres.

Siguiendo el concepto básico de este método, se puede llegar incluso hasta tener un cruce multipunto, sin embargo esta idea podría reducir el rendimiento del algoritmo genético, dado que añadiendo nuevos puntos de cruce, se va corrompiendo la información genética que los nuevos individuos puedan heredar, ya que al tener un número elevado de segmentos se pierde la esencia genética de los padres, es por ello que hay que evitar corromper dichos segmentos también llamados bloques constructivos.

2.4.3.1.2 Uniforme

El método se caracteriza por la igualdad de probabilidades que cada unos de los genes de los cromosomas padres tienen para pertenecer a los individuos en la nueva generación.

La uniformidad presente, brinda la oportunidad de tener un cierto nivel de control sobre la reproducción en los individuos de la población actual, ya que por medio de una máscara o plantilla podemos controlar las posiciones de los genes que serán heredados a los individuos que se generen.

Solo hace falta recorrer los valores binarios que la máscara tenga, e ir asignando los genes a los individuos de la próxima generación.

2.4.3.1.3 Scattered

De acuerdo a un numero binario generado aleatoriamente, se puede asignar los genes de los cromosomas padres a los nuevos individuos, según el valor obtenido.

El método tiene el inconveniente, de la posibilidad de no mantener la estructura genética de los padres y de corromper la herencia que los futuros individuos puedan tener.

2.4.3.2 Mutación

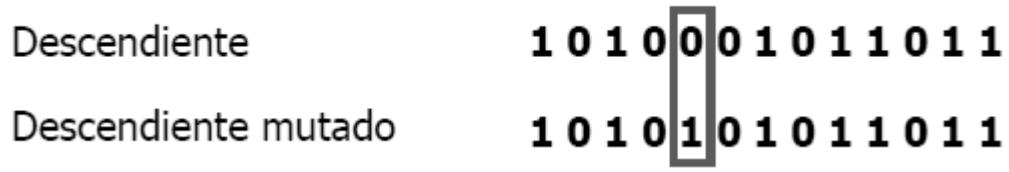


Figura 8 – Ejemplo mutación en un solo bit [18].

La mutación de un individuo provoca que algunos de sus genes cambien su valor, por uno aleatorio, la Figura 8 muestra este proceso.

La variación del valor es según el tipo de codificación, por ejemplo si se utiliza una codificación binaria bastaría con negar el bit para producir la mutación, en algunos problemas incluso la mutación puede producirse con el intercambio de los valores de un par de genes.
Así mismo existen otras opciones de mutación en codificaciones con valores reales o enteros, como incrementar o decrementar en una pequeña cantidad el valor de un gen elegido aleatoriamente.

La probabilidad de mutación con la que se trabajo en la mayoría de los casos es mínima, los valores rondan el 1 porciento, dicho valor es mínimo ya que lo que se busca es introducir nueva información genética sin provocar variaciones en los bloques constructivos.

Generalmente la mutación es empleada después de usar el cruce en la reproducción de los individuos, ya que después de generar una descendencia siempre se produce una variación de la información genética, dicha variación no tiene mayor transcendencia en la herencia genética de los hijos, pero si una gran importancia en la exploración del espacio de soluciones.

2.4.3.2.1 Bit

La mutación se produce solo en un gen seleccionado aleatoriamente, como se menciono anteriormente la variación que se puede producir dependerá de la codificación del cromosoma.

2.4.3.2.2 MultiBit

Realizando una generalización del anterior método se puede, incrementar el número de genes que modifiquen su valor, cabe recalcar que al elevar el número de genes mutados, es posible perder la herencia de los padres, modificando en demasía los bloques constructivos.

2.4.4 Evaluación

Es la función empleada para evaluar la idoneidad de un individuo determinado, de manera que evaluar la aptitud de un individuo consiste en aplicar la función para producir un resultado, o un rango, el cual mida la calidad del individuo como solución del problema.

El resultado permitirá controlar el número de selecciones, cruces, mutaciones, etc. Que se lleven a cabo.

La evaluación es determinante en la eficiencia y eficacia del algoritmo genético, porque el resultado de la función influye en la convergencia de la solución del problema, dentro de esta función de evaluación es importante incorporar las restricciones del problema, teniendo en cuenta los limites que enmarcan al problema.

Cada función de evaluación es diseñada exclusivamente para cada problema, como es de esperarse la implementación del método se convierte en un proceso complejo de realizar.

3 Desarrollo

Para diseñar una red como por ejemplo la presentada en la figura 9 con múltiples nodos, distancia extensa entre nodos, requerimientos de ancho de banda asimétricos, distintos tipos de enlaces con capacidades diferentes, etc.
Es conveniente contar con un software capaz de aproximar una solución, mediante la generación de propuestas varias, hasta encontrar la adecuada.

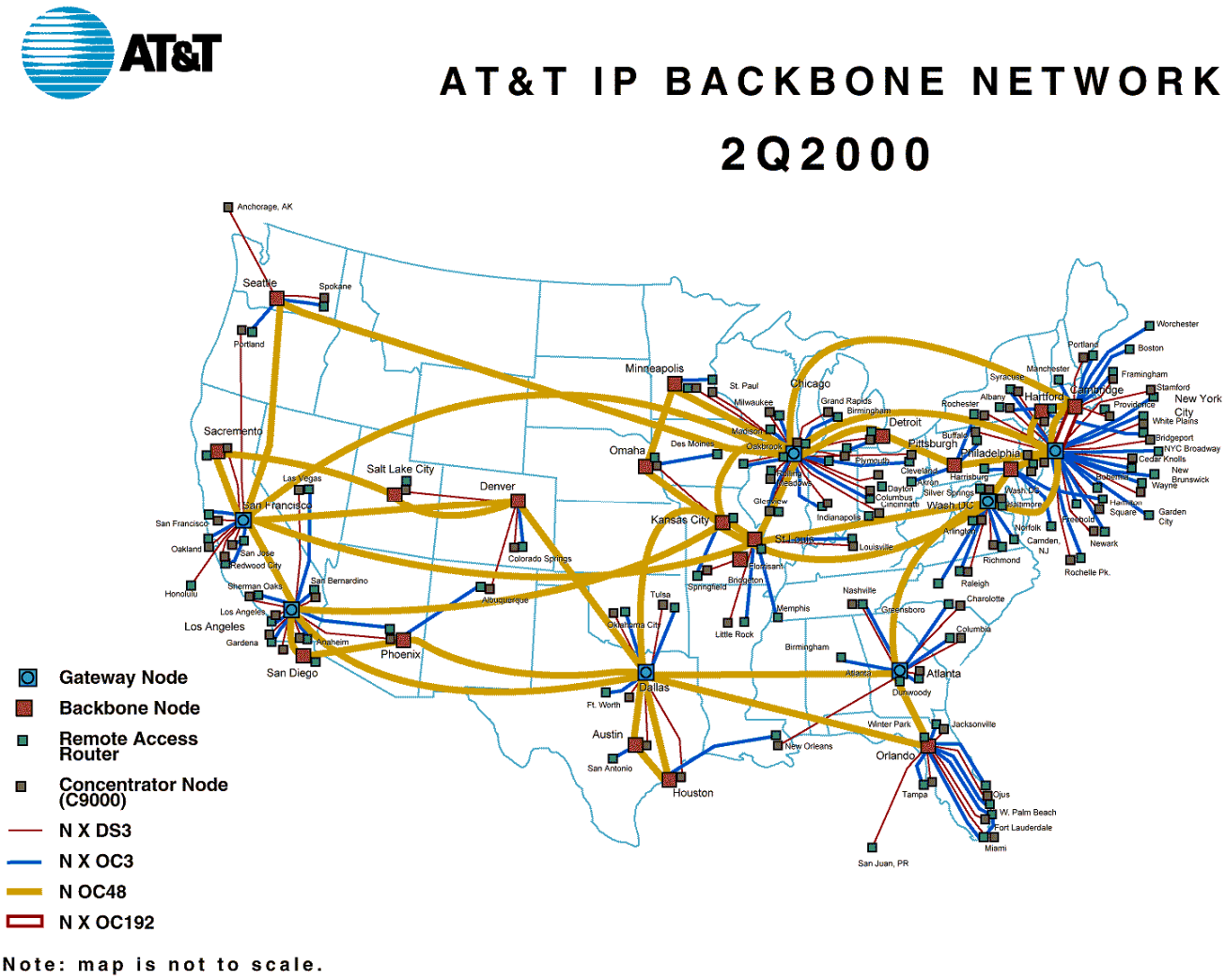


Figura 9 – Núcleo Backbone de Internet [20].

El PFC está compuesto por 6 etapas principales como muestra la Figura 10, cada una proporciona una funcionalidad importante al proyecto, para desarrollar el software capaz de diseñar redes con el mínimo costo posible.

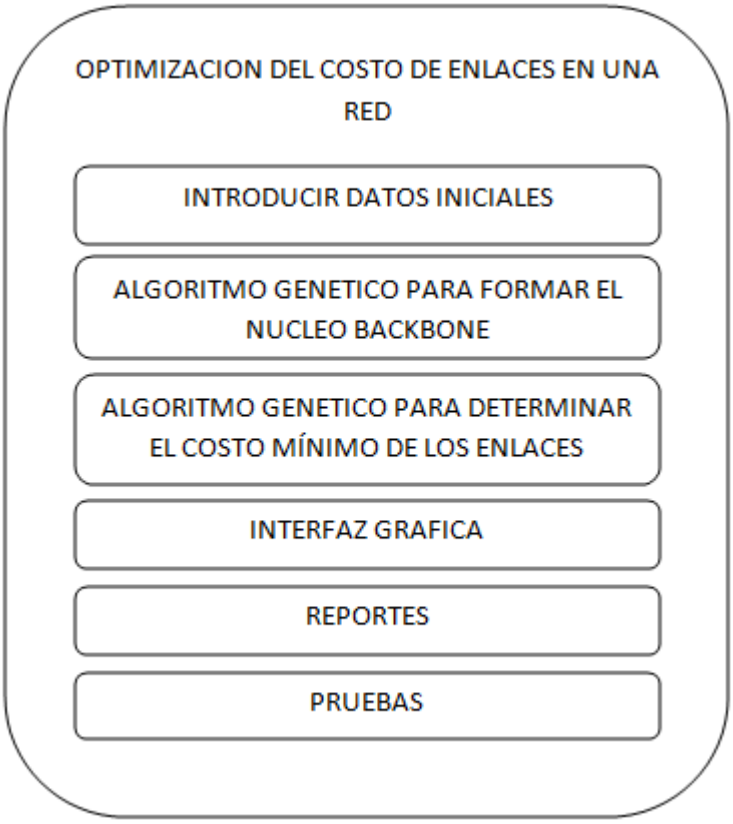


Figura 10 – Estructura del Proyecto de Fin de Carrera

Para poder comprender cada una de las etapas, se procederá a describir en detalle, tanto su funcionalidad, código e interfaz grafica asociadas a la misma.

3.1 Modulo para introducir datos

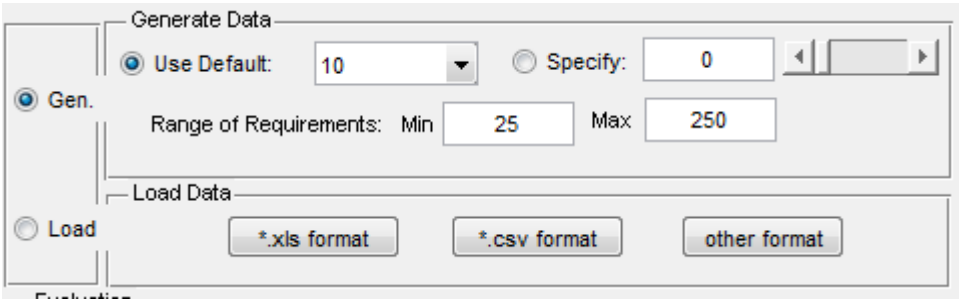


Figura 11 – Fracción de la Interfaz correspondiente a la Introducción de datos

Los datos iniciales son muy importantes ya que conforman la base de la futura solución. Para ejecutar los algoritmos genéticos, los datos iniciales, son:

- Los nombres de las ciudades o nodos
- La población o densidad de usuarios que los nodos presenten
- Las coordenadas reales de la ciudad o nodos

- La matriz de requerimientos de ancho de banda entre nodos
- La capacidad de los enlaces con los que se cuente para interconectar los nodos

El modulo obtiene los datos necesarios que representen el problema que se desee resolver, para conseguir este objetivo existen dos maneras definidas:

- Generar datos aleatorios
- Cargar datos de archivos

3.1.1 Generar Datos Aleatorios

`agGenerarDatosIniciales.m`

Es la función encargada de generar los datos, según las opciones seleccionadas por los usuarios se generaran los datos empleando técnicas de azar, como son las funciones aleatorias, así podremos conformar los datos necesarios para ejecutar los algoritmos genéticos.
Al emplear datos aleatorios, se plantea una situación hipotética, entrando al campo de la simulación, ya que los resultados obtenidos se limitan por ejemplo a comprobar el funcionamiento del software o para realizar aproximaciones del rendimiento del mismo.

`agNumRandom.m`

Es la función necesaria para generar números aleatorios entre un intervalo.

Funciones de MATLAB

```
zeros : Genera matrices con valor cero en cada elemento.
size  : Indica el tamaño de la matriz introducida, en formato
(filas,columnas).
```

Código Adicional

Para capturar las opciones del usuario, es necesario manejar los componentes de la interfaz grafica.
Como ejemplo se explicara el funcionamiento del componente, Pop-up Menu

Con la función "`get`", se captura la información seleccionada, para almacenar los valores, estos deben formar parte de la estructura de datos que MATLAB maneja por defecto, dicha estructura se denomina "`handles`".
Una vez los valores pertenecen a la estructura de datos, mediante la función "`guidata`" se debe guardar la nueva estructura.

```
%handles.vScaling
valorScaling=get(handles.cbScaling,'Value');
switch valorScaling
    case 1
        handles.vScaling=1;
    case 2
        handles.vScaling=0;
end
guidata(hObject,handles);
```

3.1.2 Cargar Datos de Archivos

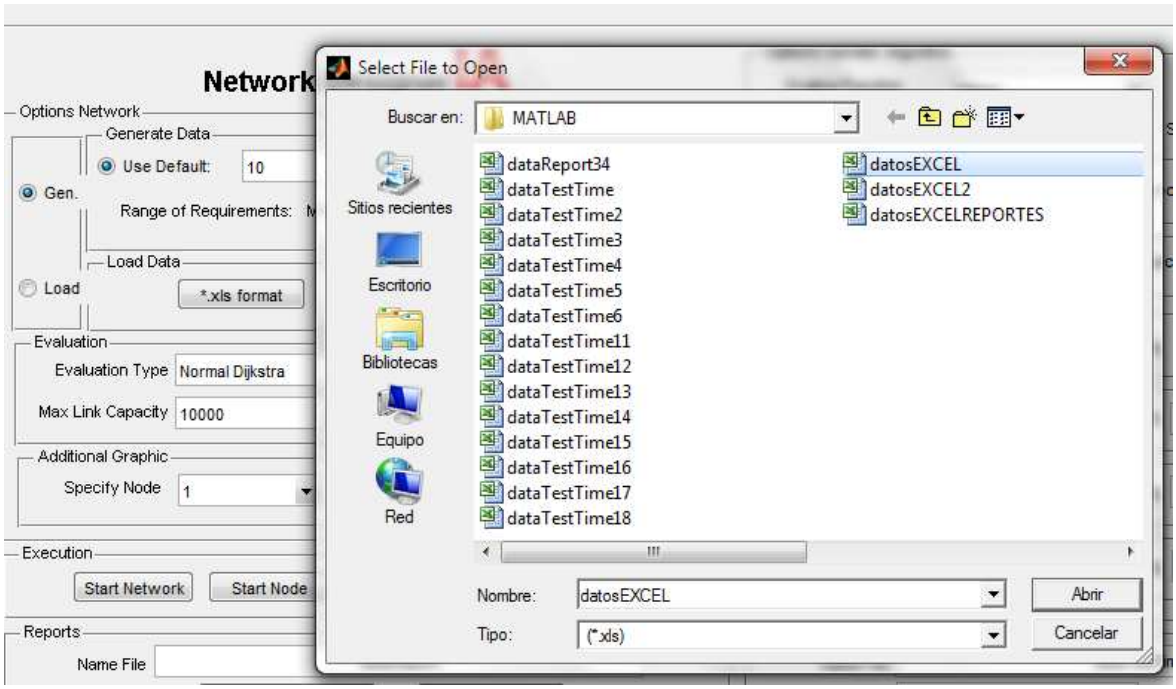


Figura 12 – Procedimiento para seleccionar el archivo para cargar datos

Para cargar los datos de un problema, es necesario seleccionar el formato del archivo que será cargado, y comprobar que el archivo mantenga un orden específico de los datos, para que el software pueda leer la información correctamente.

El formato establecido para el software es:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	12	14	3	14	10	2	5	9	14	15	3	15	14	8	0
2	12	0	3	7	14	12	14	10	1	13	14	11	12	11	6
3	10	3	0	11	1	5	2	2	13	11	5	14	1	7	6
4	12	12	4	0	8	7	10	11	12	5	11	10	3	3	8
5	14	6	9	4	0	12	5	8	11	13	14	9	3	3	5
6	13	5	12	4	14	0	6	4	5	10	8	6	13	9	9
7	14	5	12	12	6	9	0	2	2	8	12	14	3	9	8
8	1	6	3	12	5	8	3	0	9	5	10	11	11	7	2
9	4	14	3	13	9	15	2	7	0	2	14	1	12	12	13
10	2	7	5	12	7	14	4	5	3	0	3	13	9	9	3
11	13	10	6	8	7	2	4	3	4	4	0	7	2	14	14
12	8	8	6	14	6	3	12	6	4	7	2	0	3	14	14
13	9	2	4	6	12	1	2	3	10	11	10	7	0	9	5
14	11	4	11	4	6	10	12	2	14	12	8	7	7	0	5
15	8	8	12	12	10	6	12	8	6	14	13	9	10	9	0
16															
17															
18															

requerimientos | coordenadas | nombreCiudades | poblacion | capacidadEnlaces

Figura 13 – Representación de los requerimientos en formato Excel

Formato Excel: En cada Sheet se especifica la siguiente información en el orden establecido.

- Requerimientos
- Coordenadas
- Nombre de las Ciudades
- Población
- Capacidad de los Enlaces

Formato CSV: Al no contar con workbooks y sheets, como en el Formato Excel, los datos estarán separados en archivos individuales, es por ello necesario especificar en cada archivo CSV, el nombre del proyecto seguido del dato que está almacenado en el mismo.

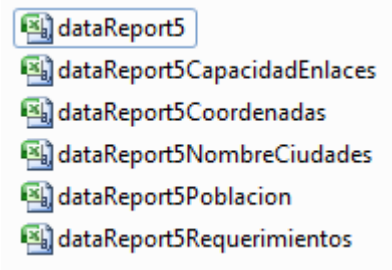


Figura 14 – Lista de los datos iniciales en formato CSV

Por ejemplo, si el proyecto a cargar se denomina Diseno1, para cargar los datos será necesario que el nombre de los archivos lleve el siguiente formato:

- Diseno1
- Diseno1Requerimientos
- Diseno1Coordenadas
- Diseno1NombreCiudades
- Diseno1Poblacion
- Diseno1CapacidadEnlaces

Código Adicional

Para cargar los datos iniciales, necesitamos seleccionar un archivo, del cual se obtendrá su PATH, con el siguiente código podemos realizar este proceso.

```
handles.fileName = uigetfile('*.csv');
if ~isequal(handles.fileName, 0)
    guidata(hObject,handles);
end
```

Luego procedemos a leer la información del archivo específico, añadiendo al PATH obtenido el nombre del dato que leeremos, luego almacenamos los datos.

```
requerimientos=strcat(fileName,'Requerimientos.csv');
m = csvread(requerimientos);
handles.requerimientosLoad=m;
```

```
guidata(hObject,handles);
```

3.2 Algoritmo Genético para determinar el núcleo de nodos backbone

Diseñar una red, puede ser una tarea fácil si el número de nodos por interconectar es mínimo, pero a medida que el numero de nodos aumenta, el grado de complejidad se incrementa.
Es por ello que en la planificación de este software se tomo la decisión de crear un algoritmo genético, el cual genere un núcleo Backbone formado por un número de nodos determinado por el usuario.
El proceso reducirá el tiempo de procesamiento al buscar una solución, porque el software trabajara con un limitado número de nodos, ya que los nodos que no formen parte del núcleo Backbone se acoplaran al nodo Backbone más cercano para ellos.

3.2.1 Generar datos iniciales

Para poder ejecutar el algoritmo genético, debemos contar con todos los datos iniciales para su funcionamiento.
Uno de los datos es la distancia existente entre los nodos, para obtener esta información utilizamos la siguiente función:

`agMatrizDistancia.m`

La función tiene como objetivo calcular la distancia existente entre dos puntos ubicados en el plano terrestre, los cuales están representados por su latitud y longitud.
Al trabajar con coordenadas reales, necesitamos una formula, la cual, pueda calcular con precisión la distancia real exacta entre dos puntos.

La fórmula empleada es la siguiente:

$$D = \text{Radio de la Tierra} * \arccos[\operatorname{sen}(\text{latitud}_a)\operatorname{sen}(\text{latitud}_b) + \cos(\text{latitud}_a)\cos(\text{latitud}_b)\cos(\text{longitud}_a - \text{longitud}_b)] \quad (1)$$

3.2.2 Generar población inicial

Se genera la primera población de cromosomas aleatoriamente de acuerdo al número de nodos Backbone que se especifico. Teniendo en cuenta que no puede existir varias veces el mismo nodo en un cromosoma el algoritmo verifica que no existan elementos repetidos al formar los cromosomas y por ende al formar la población inicial aleatoria.

`agGenerarCromosomasRedes.m`

La función es la encargada de generar la población inicial, creando cromosomas con valores aleatorios, hasta llegar al número que el usuario determino.

3.2.3 Generar población según criterio

El algoritmo ingresa en un ciclo iterativo hasta alcanzar el numero de iteraciones que el usuario determino.

Se crea una nueva población, de acuerdo a la selección del ajuste que el usuario determine:

- elitismo
- aleatorio

Si la opción elitismo es seleccionada, la nueva población mantendrá los dos mejores resultados de la anterior población, garantizando que el algoritmo convergerá hacia una solución igual o mejor que la actual. De no seleccionarse el elitismo todo el proceso será aleatorio.

3.2.4 Selección de cromosomas

Para continuar la generación de la nueva población, se necesita seleccionar los cromosomas que serán los padres, los cuales mediante métodos de cruce y mutación crearan los hijos que formaran parte de la nueva población.

Los métodos elegidos para la selección de cromosomas son:

Probabilístico: Se asigna una probabilidad a cada cromosoma, de acuerdo al valor de adaptación que el mismo presenta en relación a la solución esperada. Para luego, mediante un valor aleatorio determinar que cromosoma es seleccionado.

Para el funcionamiento del método, las siguientes funciones son desarrolladas:

`agGenerarPoblacionElitismoSeleccionProbabilidad.m`

Función principal para el método de selección probabilística.

`agGenerarRangoProbabilidadesRN.m`

Para generar el rango de probabilidades se toma en cuenta el orden descendente de la población, es decir, los primeros cromosomas serán los mejores y los últimos los peores. La función determina los rangos de probabilidad que le corresponde a cada cromosoma según el número total de cromosomas que existe en la población.

La función es utilizada solo una vez, porque son los cromosomas los que cambian de orden en la población y el rango de probabilidades se mantiene estático.

`agNumRandomReal.m`

Función que genera números aleatorios reales entre dos limites determinados.

`agPosicionRangoProbabilidadesRN.m`

La función determina a que cromosoma corresponde un rango de probabilidades, es decir al tener un valor se desea determinar en qué rango de probabilidades se encuentra y por ende a que cromosoma corresponde.

Torneo: De un grupo de cromosomas seleccionados aleatoriamente se elige al mejor de ellos, es decir, el cromosoma que tenga el valor de adaptación más alto.

Para el funcionamiento del método, las siguientes funciones son desarrolladas:

`agGenerarPoblacionElitismoSeleccionTorneo.m`

Función principal para el método de selección por torneo.

`agDeterminarGanadorTorneoRedes.m`

La función es la encargada de determinar el mejor cromosoma que será elegido de un grupo de cromosomas seleccionados aleatoriamente.

Para ello se selecciona un cromosoma, el cual es evaluado para determinar su valor de adaptación.
Tanto el cromosoma como su correspondiente valor de adaptación son almacenados en una matriz auxiliar, la cual es completada con los valores de los siguientes cromosomas hasta completar el número determinado para el grupo.

`agOrdenarDesviacionEstandarSeleccionTorneo.m`

Función para ordenar una matriz de datos.

Funciones de MATLAB

`sortrows: función para ordenar una matriz según la fila o columna seleccionada.`

Aleatorio: La selección de los cromosomas es totalmente aleatoria, sin considerar ningún tipo de método, opción, u influencia.

Para el funcionamiento del método, la siguiente función es desarrollada:

`agGenerarPoblacionRedesElitismo.m`

Función principal para el método de selección aleatoria.

3.2.5 Generar nuevo cromosomas mediante operaciones de cruce y mutación

Para crear los nuevos cromosomas a partir de los cromosomas seleccionados, se hace uso de los siguientes métodos de cruce:

`agCrucePorcentaje.m`

Porcentaje: Según un valor determinado por el usuario, se elige un punto en el que los segmentos que se genere en los padres, serán intercambiados para que los hijos hereden la información genética de ambos padres.

`agCruce2puntos.m`

2 Puntos: El método debe garantizar que exista tres segmentos en cada padre formados por los dos puntos elegidos al azar.

`agCruceScattered.m`

Scattered: Se recorre todos los genes del cromosoma padre, y mediante un valor aleatorio se determina a que hijo será insertado dicho gen.

Los métodos de cruce desarrollados emplean la siguiente función para verificar y corregir que las soluciones sean validas, es decir que un gen no se repita varias veces en un cromosoma.

`agComprobarHijoBasicoRedes.m`

Funciones de MATLAB

`round: función para redondear un valor al entero mas cercano.`

3.2.6 Evaluación mediante desviación estándar

La función de evaluación, es de mucha importancia, ya que nos permite clasificar las soluciones parciales y tener una referencia de las mejores soluciones actuales.

Dentro del presente algoritmo genético la función de evaluación consiste en obtener un cromosoma, asumir que dicho cromosoma es el núcleo Backbone de la red, y asociar los nodos restantes a los nodos Backbone más cercanos.

De esta manera se formaran grupos de nodos, los cuales serán comparados para determinar la simetría en cuanto a la cantidad de tráfico que puedan generar.

Mientras mas simétricos sean los grupos formados la solución tendrá una mejor clasificación.

`agPromedioCostoDesviacionEstandar.m`

Función principal para la evaluación de los cromosomas.

`agFormarGruposBackbone.m`

Función utilizada para formar grupos de nodos dentro de la red.

`agDeterminarConexionBackbone.m`

Función empleada para asociar un nodo, al nodo Backbone más cercano.

`agGravedadReilly.m`

Función encargada de calcular el valor correspondiente entre 2 poblaciones y la distancia entre ellas.

`agEvaluarRedesDesviacionEstandar.m`

Para determinar la simetría de los grupos formados, se emplea la técnica de la desviación estándar, aplicando la siguiente fórmula:

$$\sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

Si el resultado es menor, se interpreta que los grupos son simétricos, caso contrario si el resultado es mayor se asume que los grupos son diferentes.

Hasta aquí el funcionamiento del algoritmo genético, luego se irá repitiendo el proceso hasta concluir con el número de iteraciones determinado.

Funciones de MATLAB

`ones: crea una matriz con valores igual a uno.`

3.2.7 Gravedad de Reilly

Es una ley, la cual indica cómo se lleva a cabo la atracción entre las poblaciones y la distancia que existe entre ellas.

Según la ley dos poblaciones se atraen en proporción directa al número de habitantes de las ciudades y en proporción inversa al cuadrado de la distancia entre ellas, es decir cuántos mas habitantes posean las ciudades, mayor es la atracción, además, cuanto más cerca se encuentren entre sí, la atracción se incrementa.

$$G = \frac{m_1 m_2}{d^2}$$

Con la formula descrita podemos llevar a cabo la formación de grupos entre los nodos de la red.

3.3 Algoritmo Genético que determine el costo mínimo para interconectar en núcleo de nodos backbone

El algoritmo genético tiene como objetivo determinar los enlaces necesarios para interconectar los nodos de la red, garantizando que los requerimientos se cumplan y reduciendo al mínimo el costo que implica este proceso.
Para llevar a cabo el funcionamiento del algoritmo genético se desarrolla las siguientes funciones.

`agProgramaPrueba.m`

Función principal para ejecutar el algoritmo genético.

La siguiente función que fue descrita en detalle anteriormente, y será utiliza también en el presente algoritmo genético, ya que el método para calcular la probabilidad para cada cromosoma es la misma en ambos algoritmos genéticos.

`agGenerarRangoProbabilidadesRN.m`

3.3.1 Generar población inicial

Con la siguiente función creamos la población inicial, generando aleatoriamente valores que representan los enlaces que existen para interconectar los nodos dentro de la red.
Si se cuenta con los valores que representan los enlaces, se genera números aleatorios que pertenezcan a dicho conjunto para formar soluciones validas.

```
agGenerarCromosomasEnlaces.m
```

Para determinar el número de genes en un cromosoma, es necesario tomar en cuenta que si se trabaja con todas las posibles combinaciones existentes para interconectar los nodos de la red, el proceso será ineficiente, ya que dicho valor se incrementa factorialmente de acuerdo al número de nodos Backbone con los que se ejecute el algoritmo.

Es por ello necesario reducir la cantidad de genes por cromosoma, para superar esta dificultad, se emplea la siguiente fórmula para calcular el número de genes de un cromosoma según la cantidad de nodos Backbone.

tamaño = ((Nodos Backbone * Nodos Backbone) - Nodos Backbone) / 2

3.3.2 Selección de cromosomas

La funcionalidad de los métodos de selección empleados en el anterior algoritmo genético, son utilizados de igual manera en el presente algoritmo.
A pesar de que los métodos sean los mismos, no se debe asumir que la funcionalidad del algoritmo genético también lo sea, ya que existen diferencias en métodos posteriores como ser la evaluación.

```
agGenerarPoblacionElitismoEnlaceSeleccionProbabilidad.m
agGenerarPoblacionElitismoSeleccionTorneoEnlace.m
agGenerarPoblacionEnlaceElitismo.m
```

3.3.3 Generar población según criterio

De la misma manera que en el algoritmo genético anterior, según la decisión del usuario podremos mantener los mejores cromosomas de una población para que puedan continuar en las próximas generaciones.

3.3.4 Generar nuevo cromosoma mediante operaciones de cruce y mutación

La funcionalidad de los métodos de reproducción para el algoritmo genético es similar al anterior algoritmo descrito.

```
agCruceEnlacePorcentaje.m
agCruce2puntosEnlace.m
agCruceScatteredEnlace.m
agMutacionEnlacePorcentaje.m
```

3.3.5 Evaluación según valor asignado

Para evaluar un cromosoma y determinar su valor de adaptación empleamos el algoritmo de Dijkstra para encontrar los caminos más cortos entre los nodos, adicionalmente podemos tomar en cuanto otros parámetros como la simetría del trafico utilizado por los nodos, el tiempo de transmisión de un paquete, etc. Según los requerimientos del usuario.

`agPromedioCostoEnlaces.m`

Desde la función principal para evaluar los cromosomas, podemos llamar a la función de evaluación seleccionada, incluso existe opciones para seleccionar varios tipos de evaluación y combinar los parámetros, los tipos de evaluación desarrollados para el software son:

`agEvaluarEnlacesDijkstra.m`

La primera función de evaluación considera solo el algoritmo de Dijkstra como único parámetro para evaluar a los cromosomas.
Para evaluar los cromosomas primero se emplea el algoritmo de Dijkstra para generar los caminos más cortos de un gen a los demás genes del cromosoma.
A continuación calculamos los requerimientos que cada gen necesita dentro de la red, para garantizar el ancho de banda que la red necesita.
Mediante ciclos iterativos y utilizando la matriz de caminos cortos que genera el algoritmo de Dijkstra podemos obtener los requerimientos globales de la red.
Con los requerimientos globales de la red, y con los datos de la capacidad de los enlaces proporcionados por el usuario, podemos determinar que cromosoma es el mejor, realizando una sumatoria de la diferencia de la capacidad de un enlace con el requerimiento correspondiente para dicho enlace.
De manera que si el valor obtenido es mínimo, podemos concluir que es una mejor respuesta que un cromosoma con un valor mayor, ya que los enlaces que el algoritmo genético propone se adaptan mejor a los requerimientos de la red.
`agEvaluarEnlacesDijkstraRedHomogenea.m`

Al ser una modificación del primer método de evaluación, mantiene la estructura principal, sin embargo presenta cambios en el criterio para determinar que cromosomas son mejores.
Para obtener una clasificación de los cromosomas, se utiliza la desviación estándar para determinar si el tráfico utilizado por los enlaces mantiene un porcentaje similar entre ellos.
Es decir, mientras el porcentaje de uso entre los enlaces sea similar, el cromosoma presentara un mejor valor de adaptación dado que la red es considerada homogénea, caso contrario si el porcentaje de uso de los enlaces, presenta variaciones el valor de adaptación ira empeorando según la importancia de las variaciones.

`agEvaluarEnlacesDijkstraDelayMM1.m`

De igual manera que en el anterior caso, el criterio de clasificación de los usuarios es diferente.
Se considera un mejor cromosoma aquel que presente el tiempo total mínimo, en que un paquete recorre un enlace entre los nodos respectivos.
Utilizando la formula generalizada del M/M/1 con todos los enlaces existentes en la red, es posible obtener una clasificación de los cromosomas en una población, dado que los

cromosomas que presenten un tiempo total menor serán considerados mejores que aquellos cromosomas con un tiempo mayor.

`calcularDelayMM1.m`

La función calcula el tiempo de retardo que un paquete presenta al atravesar un enlace entre dos nodos, utilizando una modificación de la fórmula del M/M/1.

`agOrdenarCromosomasEnlaces.m`

La función es utilizada para ordenar los cromosomas según su valor de adaptación, en el algoritmo los mejores cromosomas tendrán un valor de adaptación mínimo, ya que lo que se busca es minimizar el costo de la red, este costo puede ser la distancia, dinero, tiempo de retardo, etc.

Funciones de MATLAB

```
sort: función que ordena los valores de un vector, según los
requerimientos en forma ascendente o descendente.
```

Código Adicional

`agAlgoritmoDijkstra.m`

Para generar la matriz de caminos cortos mediante el algoritmo de Dijkstra es necesario comprender algunos parámetros utilizados en el proceso:

- `numero de nodo : valor que identifica al nodo.`
- `distancia acumulada: distancia total del recorrido entre dos nodos.`
- `nodo procedencia : numero del nodo previo en el recorrido actual.`
- `etiqueta : valor utilizado para identificar a los nodos que fueron parte de un cálculo.`
- `permanente : valor utilizado para identificar a los nodos que fueron tomados en cuenta por el algoritmo.`

Siendo el nodo seleccionado el nodo inicial, comenzamos el proceso con una iteración que recorrerá todos los nodos de la red.
Dentro de la iteración, para el nodo actual se obtiene los nodos temporales con los que existe un enlace entre ellos.

Se procede a sumar la distancia acumulada correspondiente entre el nodo actual y cada uno de los nodos temporales, con los que existe un enlace, excepto los nodos que se denominan permanentes, es decir los nodos que fueron tomados en cuenta por el algoritmo de Dijkstra.

Se escoge el nodo con el que la distancia acumulada entre el nodo actual y el nodo temporal sea menor, procediendo a llenar en la matriz de Dijkstra, el nodo de procedencia que corresponde al nodo actual y activar la etiqueta, correspondiente a los nodos temporales, ya que mediante esta opción se informa que dichos nodos fueron parte de un cálculo previo y los mismos contienen valores temporales.

Como el nodo actual ya ha sido tomado en cuenta, pasa a denominarse nodo permanente, es decir ya no será tomado en cuenta en las próximas iteraciones.

Ahora solo de los nodos cuya etiqueta esta activada se elige el nodo que tenga la menor distancia acumulada.
Este proceso se repite hasta completar el cálculo con todos los nodos existentes, es decir hasta que todos los nodos sean permanentes.

Si se da el caso en que los nodos que pueden ser elegidos para ser el próximo permanente, tienen el mismo valor de la distancia acumulada, se escogerá aleatoriamente a uno de ellos.

En el caso de que en un cálculo, un nodo temporal contenga datos llenados previamente, estos datos solo podrán ser reemplazados por nuevos datos que mejoren la distancia acumulada actual, ya que lo que se busca es minimizar dicha distancia, caso contrario los datos temporales se mantienen descartando los datos del cálculo actual.

`agNodoEnlaces.m`

Función que obtiene para un nodo, todos los nodos en los que existe un enlace que los une.

`agRutaNodosDijkstra.m`

Función que obtiene mediante el uso de la matriz Dijkstra el camino más cortó entre dos nodos.

3.3.6 Algoritmo Dijkstra

El algoritmo nos permite determinar la distancia más corta de un nodo específico, con todos los demás nodos miembros de la red.
Aplicando este método a todos los nodos, podremos tener la información necesaria para que el tráfico de red, pueda dirigirse a través del camino más corto, lo cual implica una reducción de costos y un paso importante para determinar la solución más optima al problema.

3.3.7 Redes Homogéneas

La función busca encontrar una red, en la cual los enlaces estén utilizados proporcionalmente, es decir, según la capacidad de cada enlace miembro de la red, estos deben utilizar un porcentaje similar, de manera que la red sea homogénea.
Para encontrar redes homogéneas se calcula el tráfico que utiliza un enlace, y con la capacidad del enlace se puede determinar, el porcentaje que se está utilizando.
Luego de aplicar este procedimiento a todos los enlaces se puede determinar qué red es más homogénea con la ayuda de la desviación estándar.

3.3.8 M/M/1

Fórmula para calcular los tiempo dentro de una cola de espera, el tiempo que el sistema tardara en procesar todas los clientes, el tiempo que los clientes esperaran en la cola de espera, el número de usuarios que están en la cola de espera, el número de usuarios que están en el sistema.

Haciendo una generalización de esta fórmula y determinando algunos parámetros, esta fórmula puede ser utilizada para determinar el tiempo que tarda un paquete en recorrer un enlace, y se convierte en la base para determinar redes donde el tiempo que tarda un paquete en recorrer un enlace, es similar para todos los enlaces que formen parte de la red.

3.3.9 Delay Time

Es el término utilizado para llamar al tiempo que tarda un paquete en llegar de un nodo a otro. Para calcular el Delay Time hace falta emplear una formula, la cual tenga en cuenta la capacidad de los enlaces, el trafico que circule por los enlaces, el tamaño del paquete, etc. Una de estas formulas y de las más empleadas es la M/M/1.

3.4 Interfaz Grafica para presentar los resultados

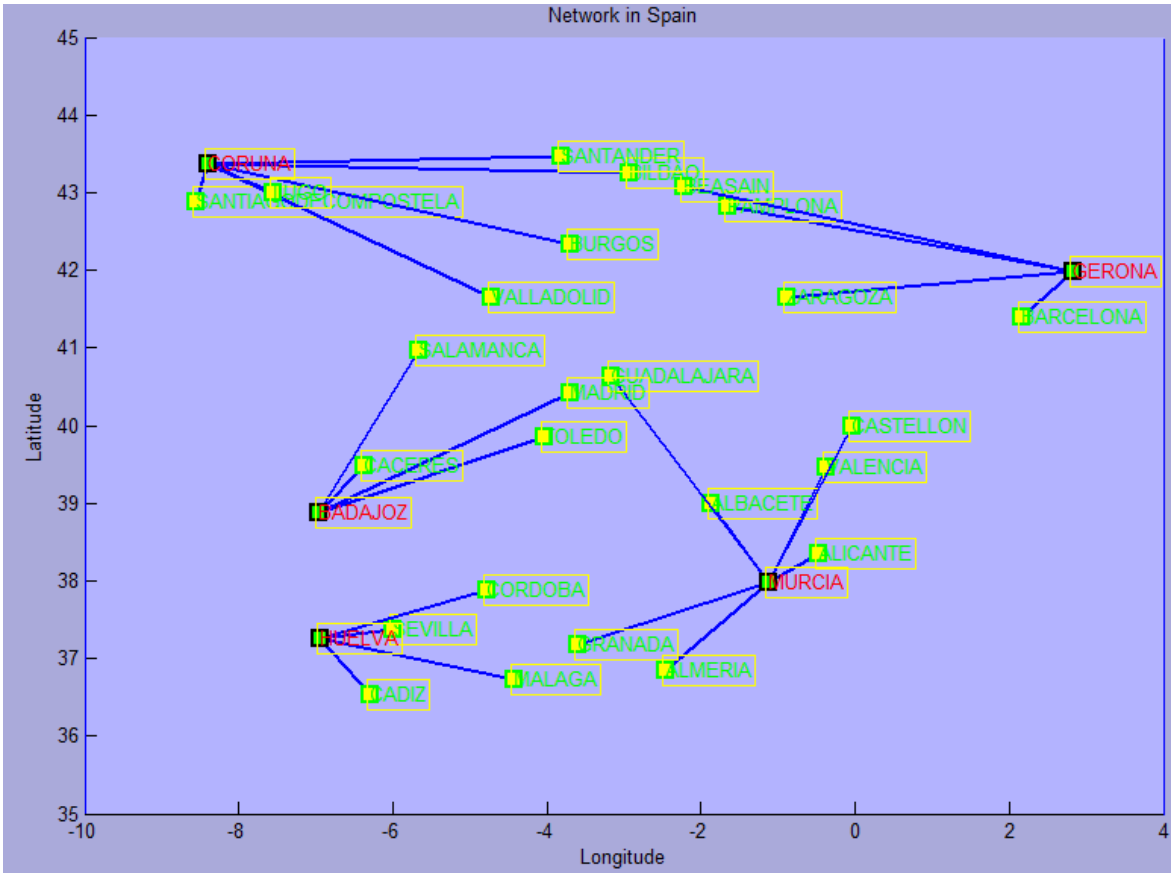


Figura 15 – Representación grafica de la formación del núcleo Backbone de una red

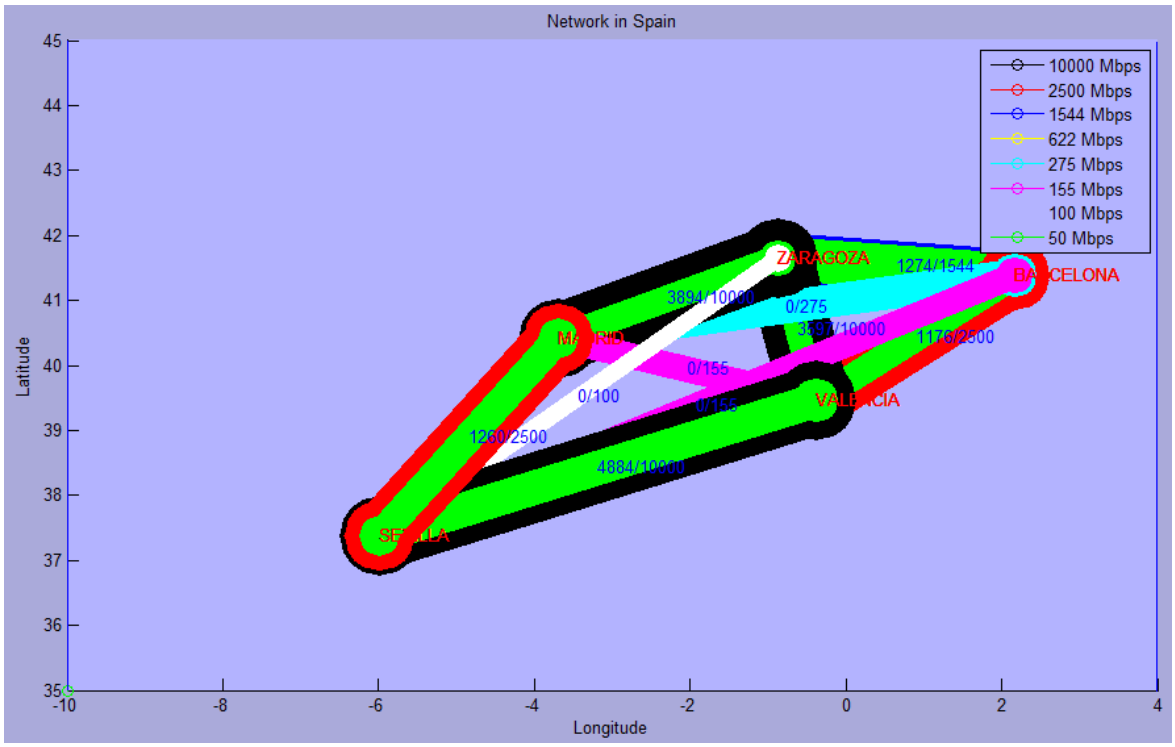


Figura 16 – Representación grafica de la conexión entre nodos

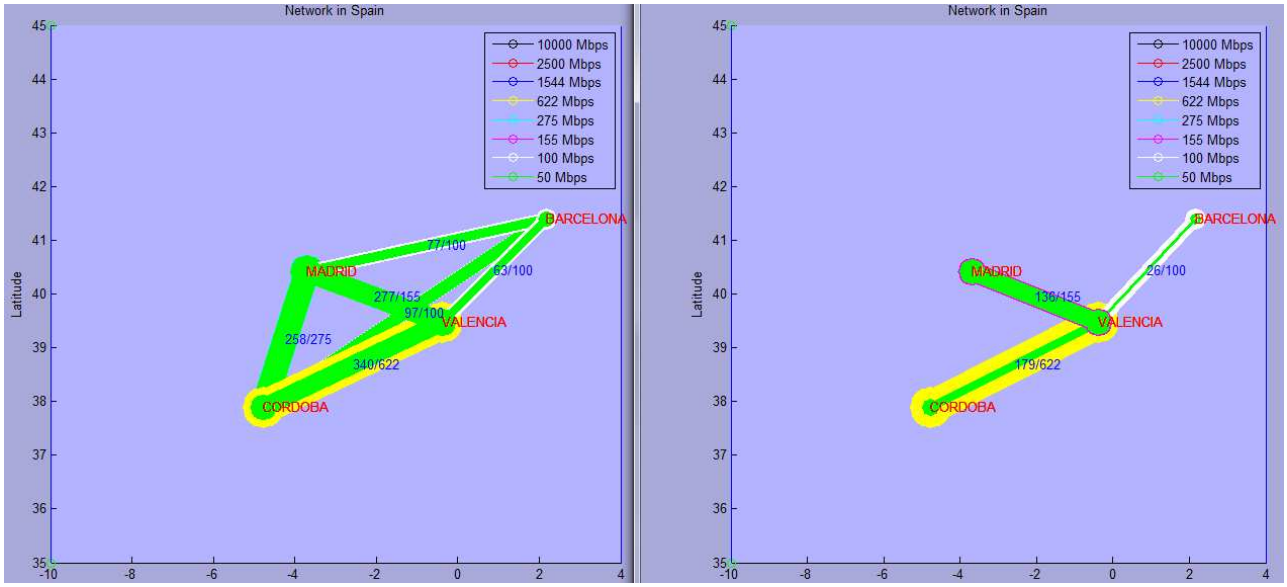


Figura 17 – Representación grafica principal de la conexión entre nodos y el grafico parcial para un nodo especifico

A pesar de ser MATLAB una herramienta especialista en matrices y vectores, y manejar la mayoría de sus resultados en formato consola, también cuenta con las funciones necesarias para crear interfaces graficas y presentar los resultados de una manera amigable y fácil de usar.

Así mismo cuenta con un IDE interno el cual facilita el diseño de la interfaz grafica.

Para programar las funciones y controlar los componentes de la interfaz grafica, por defecto MATLAB propone el desarrollo siguiendo la metodología estructurada, aunque tiene soporte para la programación orientada a objetos, el cual implica una mayor profundización en su funcionamiento.

`agDibujarGruposBackBone.m`

Función principal para generar el grafico que represente la formación del nucleo Backbone de la red.

`agDibujarLineaGrupo`

Empleando funciones de MATLAB, grafica una línea entre dos puntos específicos según las coordenadas establecidas.

`agDibujarLegend.m`

Función utilizada para insertar en el grafico las características de los enlaces disponibles para interconectar los nodos de la red.

`agDeterminarColorEnlace.m`

Función utilizada para determinar el color que un tipo de enlace tiene asignado, de esta manera podemos generar un grafico más explicativo y fácil de comprender.

`agPruebaDibujo.m`

Función principal para generar el grafico que represente la conexión entre los nodos.

`agGetLatitudLongitud.m`

Obtiene las coordenadas de un nodo, dicha información posibilita ubicar los nodos dentro del grafico.

`agDibujarLineaEnlace.m`

Empleando funciones de MATLAB, grafica una línea entre dos puntos específicos según las coordenadas establecidas.
Así mismo, determina la proporcionalidad de la línea en el grafico, y el color que representa el tipo de conexión.

`agDibujarCaracteristicasEnlace.m`

Función encargada de introducir en el grafico la información referente a la capacidad de un enlace, y el valor utilizado de dicha capacidad.

`agDibujarNombreEnlace.m`

Función encargada de introducir en el grafico el nombre de los nodos.

`agPruebaDibujoParcial.m`

Función encargada de graficar la información referente a un nodo en específico, es decir en el grafico generado se presentara las conexiones, información, etc. de un nodo seleccionado por el usuario.

Funciones de MATLAB

```
figure   : crea una nueva ventana con un lienzo listo para graficar.
clf      : elimina los gráficos existentes en el lienzo.
hold     : permite mantener o eliminar los gráficos existentes en el
mismo.
whitebg  : utilizado para cambiar el color del background del grafico.
plot     : comando principal, utilizado para graficar la información
ingresada, en varios formatos, dimensiones, etc.
legend   : permite insertar en el grafico información adicional.
xlabel   : utilizado para nombrar al eje x de un plano cartesiano.
ylabel   : utilizado para nombrar al eje y de un plano cartesiano.
title    : utilizado para colocar un titulo al grafico.
text     : empleado para colocar texto en el grafico.
```

3.5 Modulo para generar reportes

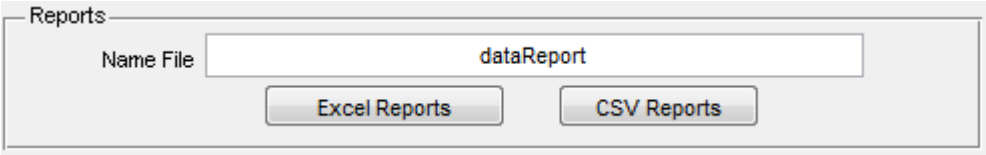


Figura 18 – Fracción de la Interfaz correspondiente a los Reportes

El modulo es encargado de exportar las soluciones, al formato que el usuario crea conveniente, para ello existen dos formatos establecidos, los cuales cubren las plataformas actuales como ser privativas y las de software libre.

3.5.1 Formato Excel

Cubriendo los requerimientos de la plataforma Windows, los resultados obtenidos pueden ser exportados al formato Excel, el cual es ampliamente utilizado por las grandes bondades y la facilidad que presenta al momento de manejar estructuras de datos como ser matrices y vectores.
Basándose en workbooks y sheets pueden organizarse los datos que serán exportados.

`agReportes.m`

La función es encargada de generar los reportes en formato Excel, los datos almacenados obtenidos como solución al problema son añadidos a un archivo Excel secuencialmente.

Cada dato es almacenado en un sheet, formando así, un workbook el cual será el archivo reporte que se identifique con el nombre establecido por el usuario.

Funciones de MATLAB

`xlswrite : función encargada de crear el archivo en formato Excel.`

3.5.2 Formato CSV

Para las plataformas Unix, se emplea el formato CSV (comma-separated values) el cual representa los datos en formato texto en donde los datos están separados por comas o punto y coma, de esta manera se pueden exportar matrices y vectores de una manera fácil.

`agReportesCSV.m`

La función es encargada de generar los reportes en formato CSV, los datos obtenidos como solución al problema son almacenados independientemente, es decir cada dato en un archivo individual.
Siguiendo dicho proceso evitamos problemas de configuración, ya que al no tener conocimiento de la longitud de los datos que se almacenaran, es conveniente almacenar los datos individualmente, para poder contar con el orden y organización adecuado.

Funciones de MATLAB

```
strcat   : función utilizada para concatenar Strings.
csvwrite : genera un archivo en formato CSV, en el cual se almacenan los
datos correspondientes.
```

La información que forme parte del reporte, es seleccionada según la importancia que implica dentro de la solución del problema.

En el presente software, se selecciono los siguientes datos para ser exportados:

- `'requerimientos'`
- `'coordenadas'`
- `'nombreCiudades'`
- `'poblacion'`
- `'capacidadEnlaces'`
- `'requerimientosFormados'`
- `'vecNodosBackbone'`
- `'poblacionFinal'`
- `'vecRespuesta'`
- `'tiempoEjecucion'`

Con los datos exportados, se tiene una visión clara sobre la solución obtenida.

4 Gestión del Proyecto

La gestión del proyecto está dividida en 5 tareas o iteraciones, las cuales representar hitos importantes dentro del proyecto.

Las tareas son:

ACTIVIDAD	Tiempo (semanas)	PRODUCTO
TAREA I	2	Modulo para introducir datos
TAREA II	3	Algoritmo Genético para determinar las terminales principales y formar grupos
TAREA III	3	Algoritmo Genético para determinar el costo mínimo para interconectar las terminales principales
TAREA IV	2	Interfaz Grafica para presentar los resultados
TAREA V	2	Modulo para generar reportes

De manera que podemos gestionar cada tarea por separado, y llevar un control de la misma.

TAREA I

El primer paso por realizar dentro del proyecto es contar con la información necesaria para poder llevar a cabo la ejecución del software, ya sea generando los datos como si se tratase de una simulación, o leyendo el escenario que el usuario proporcione.

Para esto necesitamos completar las siguientes sub-tareas:

- Programar: Generar la matriz de requerimientos (código + pruebas unitarias)
- Programar: Generar las capacidades de los enlaces (código + pruebas unitarias)
- Programar: Generar las coordenadas y nombres de los nodos (código + pruebas unitarias)
- Programar: Almacenar los datos generados (código + pruebas unitarias)

En la primera fase, las sub-tareas no conllevan una complejidad mayor, simplemente se trata de investigar sobre la programación en MATLAB y asegurar que los datos son correctos y validos para pasarlos a las siguientes fases.

Como se puede apreciar en el siguiente grafico, la estimación del tiempo para esta fase, es muy elevada ya que al principio de la etapa se nota una diferencia mínima entre el trabajo realizado y el estimado. Diferencia que va aumentando con el transcurso de los días, ya que las sub-tareas se llevan a cabo en poco tiempo, característica positiva para el proyecto.

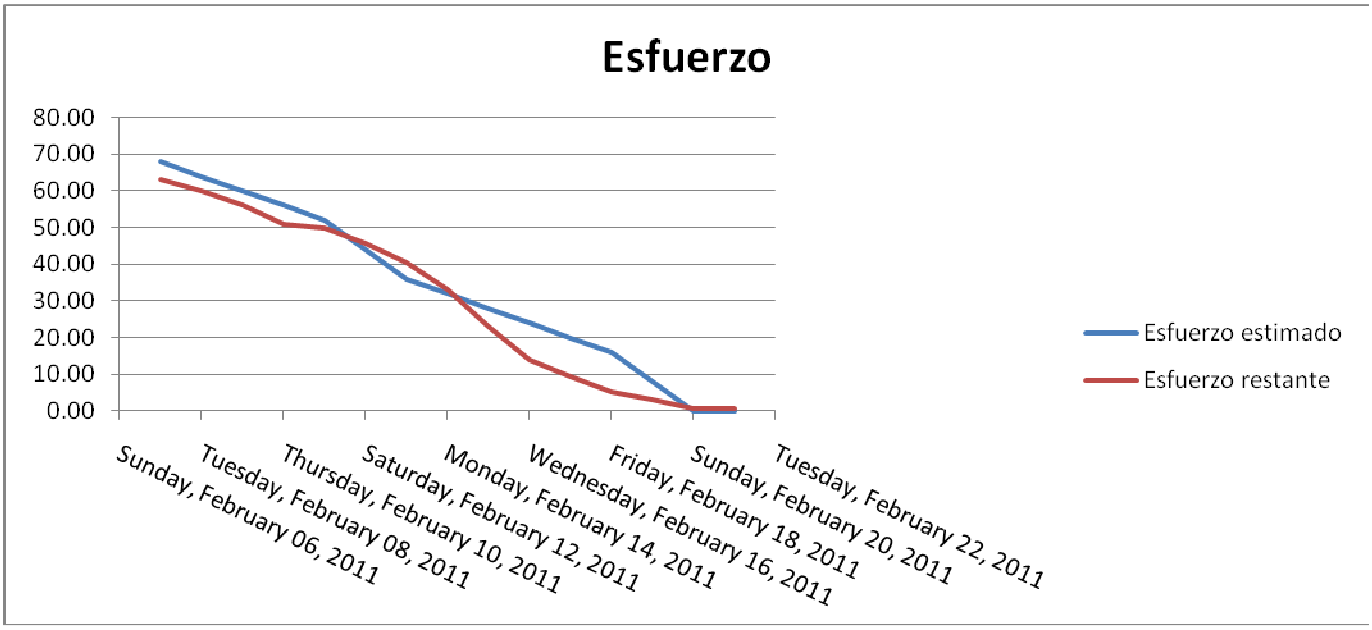


Figura 19 – Representación grafica del esfuerzo realizado y el esfuerzo estimado en la TAREA I

TAREA II

Es una de las más importantes, ya que forma parte del núcleo del proyecto, porque el desarrollo del algoritmo genético implica trabajar en el futuro con menos datos, y por consiguiente reduce el tiempo de ejecución.

El desarrollar un algoritmo genético, representa un mayor esfuerzo, ya que no existe un framework que resuelva este problema en pocas líneas de código, el trabajo por realizar empieza literalmente desde cero, y va evolucionando con el paso del tiempo.

Dependiendo de la complejidad con la que se quiera contar, es decir, codificación, inicialización, selección, reproducción, evaluación, etc. El tiempo que se necesitara para poder tener un producto aceptable incrementara exponencialmente.

Para esto necesitamos completar las siguientes sub-tareas:

- Programar: Generar población de cromosomas para agrupar los nodos(código + pruebas unitarias)
- Programar: Generar números ramdomicos (código + pruebas unitarias)
- Programar: Generar nueva población en base a una probabilidad (código + pruebas unitarias)
- Programar: Generar nueva población en base a un torneo (código + pruebas unitarias)
- Programar: Generar nueva población aleatoriamente (código + pruebas unitarias)
- Programar: Generar rango de probabilidades(código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a un cruce normal, de 2 cromosomas padres (código + pruebas unitarias)

- Programar: Generar nuevo cromosoma hijo en base a un cruce en dos puntos, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a un cruce scattered, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a una mutación normal, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Determinar ganador del torneo (código + pruebas unitarias)
- Programar: Evaluar y Ordenar la nueva población según elitismo (código + pruebas unitarias)
- Programar: Formar grupos Backbone (código + pruebas unitarias)
- Programar: Evaluar mediante gravedad de Reilly (código + pruebas unitarias)

Como se puede apreciar en el siguiente grafico, la estimación del tiempo para esta fase, es relativamente acertada, ya que al principio de la etapa se nota una diferencia entre el trabajo realizado y el estimado, al inicio de la tarea existía la susceptibilidad de si el tiempo era el suficiente para realizar dicha tarea. Pero con el pasar de los días, se pudo tener avances significativos los cuales dieron la posibilidad de concluir con la tarea satisfactoriamente.

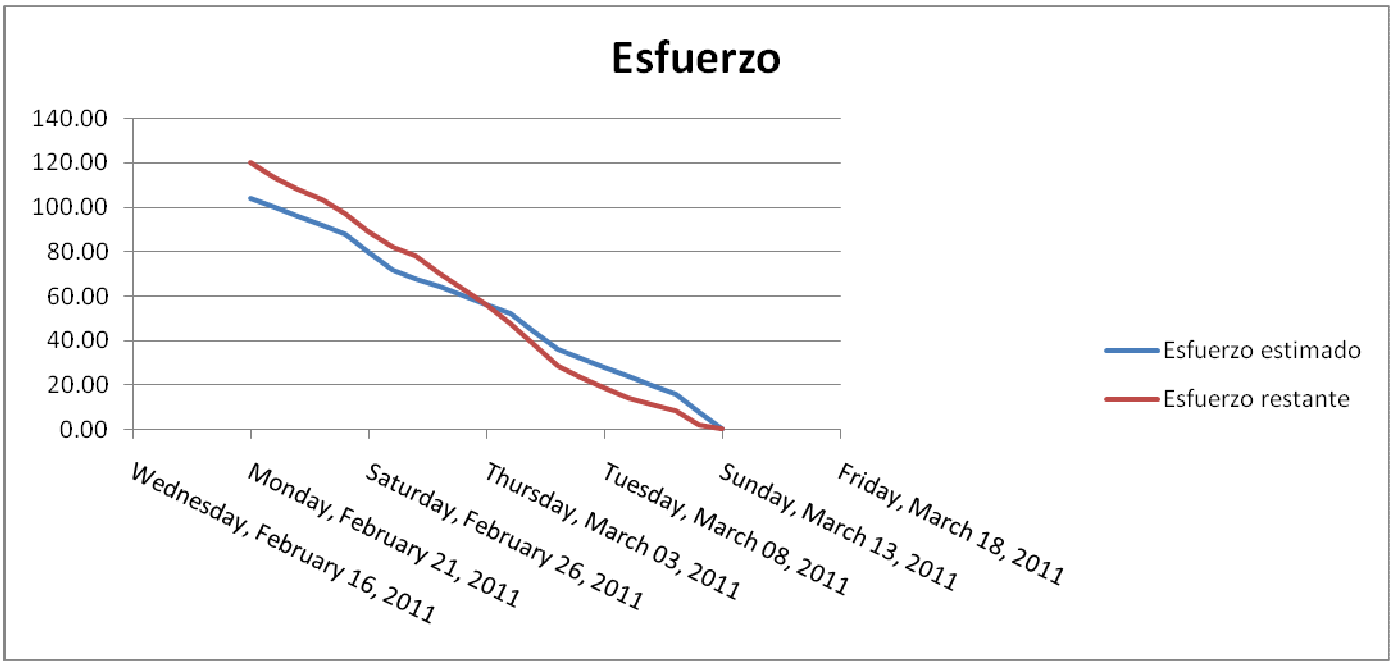


Figura 20 – Representación grafica del esfuerzo realizado y el esfuerzo estimado en la TAREA II

TAREA III

Es la tarea principal dentro del proyecto, ya que las respuestas que nos brinde este algoritmo genético se convertirán en la solución a los problemas planteados. Es por ello que el desarrollo del código del algoritmo genético tiene una prioridad alta.

Al igual que la anterior tarea, conlleva varias sub-tareas por realizar, las cuales son:

- Programar: Generar población de cromosomas para interconectar los nodos(código + pruebas unitarias)
- Programar: Generar números ramdomicos (código + pruebas unitarias)
- Programar: Generar nueva población en base a una probabilidad (código + pruebas unitarias)
- Programar: Generar nueva población en base a un torneo (código + pruebas unitarias)
- Programar: Generar nueva población aleatoriamente (código + pruebas unitarias)
- Programar: Generar rango de probabilidades(código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a un cruce normal, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a un cruce en dos puntos, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a un cruce scattered, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Generar nuevo cromosoma hijo en base a una mutación normal, de 2 cromosomas padres (código + pruebas unitarias)
- Programar: Determinar ganador del torneo (código + pruebas unitarias)
- Programar: Evaluar y Ordenar la nueva población según elitismo (código + pruebas unitarias)
- Programar: Clasificar cromosomas según algoritmo Dijkstra (código + pruebas unitarias)
- Programar: Clasificar cromosomas según algoritmo Dijkstra para redes Homogéneas (código + pruebas unitarias)
- Programar: Clasificar cromosomas según algoritmo Dijkstra controlando el delay del trafico generado en la red (código + pruebas unitarias)

Como se puede observar existe sub-tareas similares entre las iteraciones 2 y 3. Pero el hecho de que compartan un mismo identificador, no implica que sean lo mismo.

Se cree que al desarrollar un algoritmo genético, se puede reutilizar el código para generar otros algoritmos, esta idea es errada, ya que si bien es cierto que la estructura de un algoritmo genético es la misma para cualquier implementación, las funciones que un algoritmo genético tiene son únicas, porque son especificas para un problema determinado.

En el siguiente grafico se puede apreciar que el trabajo realizado es superior al estimado, debido a que el desarrollo del código implica entender las nuevas funciones necesarias y como adaptarlas a la estructura ya conocida de un algoritmo genético.

Aproximadamente hasta la segunda semana, se estimaba que el tiempo para terminar dicha tarea era insuficiente, ya que la tendencia nos indicaba que se necesitaría más tiempo, lo cual implica alargar los plazos del proyecto.

Sin embargo a raíz, de un gran salto en el desarrollo, se puedo volver a la estimación inicial, salvando los plazos establecidos, y obteniendo un producto, aceptable, el cual con la evolución del proyecto podría ir mejorando en algunos detalles, ya que la base del mismo esta totalmente desarrollada.

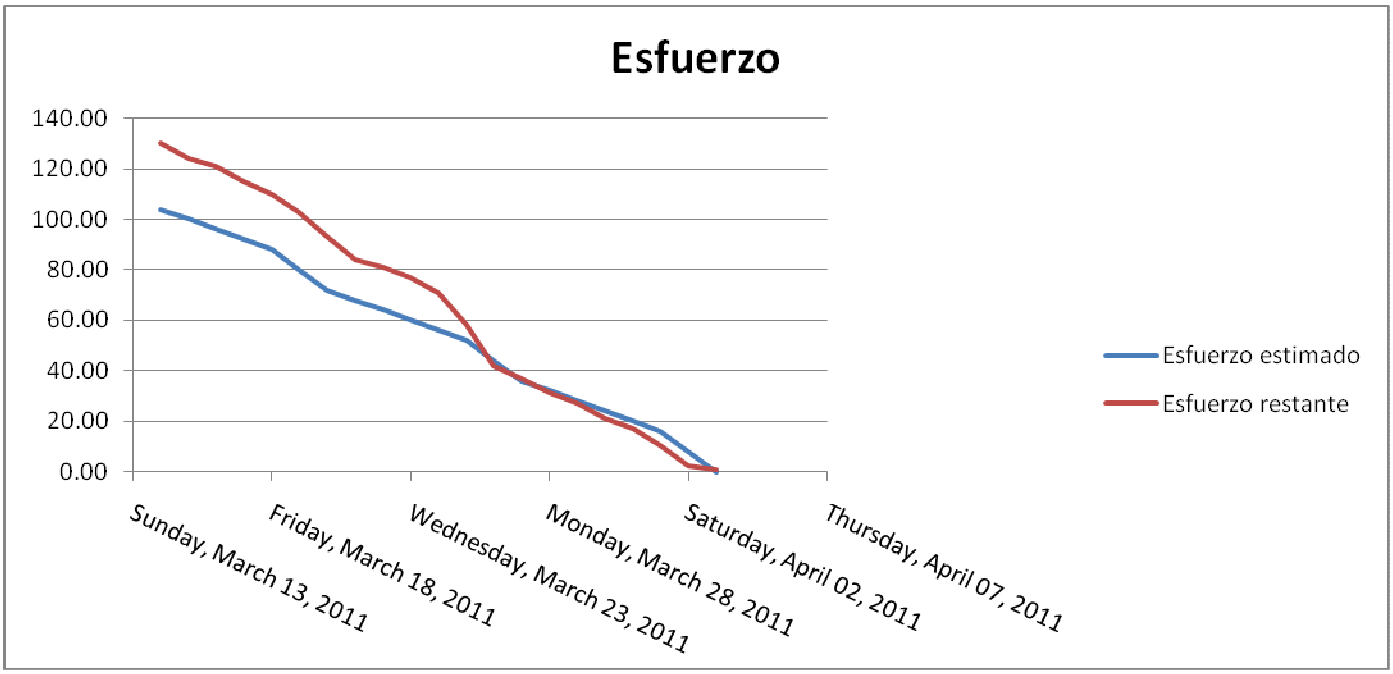


Figura 21 – Representación grafica del esfuerzo realizado y el esfuerzo estimado en la TAREA III

TAREA IV

Los resultados obtenidos por los algoritmos genéticos, hasta el momento solo están presentados a través de una consola, si bien es cierto que esto es aceptable, es necesario presentar los resultados en una interfaz grafica, amigable, y con toda la información necesaria para comprender la información que se genero.

En este momento del proyecto, podemos expresar que la parte complicada fue superada, ya que la interfaz grafica solo presenta algunas sub-tareas, las cuales, no conllevan una complejidad significativa, aunque si se debe afrontar varios problemas de errores, los cuales aparecen al momento de graficar los resultados, producto de la poco práctica para manejar algunas funciones de MATLAB.

Las sub-tareas a realizar son:

- Programar: Generar una ventana principal con la información de los enlaces(código + pruebas unitarias)
- Programar: Graficar los enlaces entre nodos según su capacidad (código + pruebas unitarias)
- Programar: Mostrar la capacidad del enlace utilizada/total (código + pruebas unitarias)
- Programar: Mostrar los nombres de las ciudades de los nodos pertenecientes al grafico (código + pruebas unitarias)
- Programar: Generar una ventana adicional con la información del nodo especifico(código + pruebas unitarias)
- Programar: Graficar los enlaces entre nodos según su capacidad del nodo especifico (código + pruebas unitarias)

- Programar: Mostrar la capacidad del enlace utilizada/total del nodo especifico(código + pruebas unitarias)
- Programar: Mostrar los nombres de las ciudades conectadas al nodo especifico(código + pruebas unitarias)

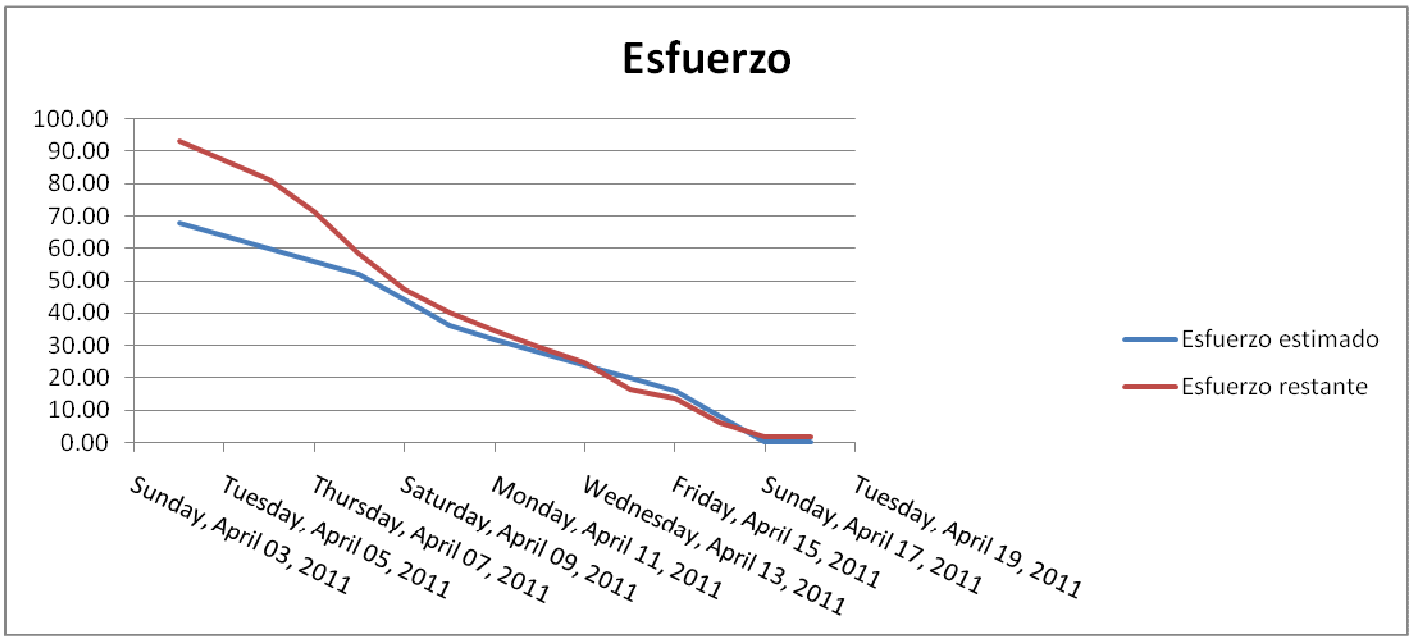


Figura 22 – Representación grafica del esfuerzo realizado y el esfuerzo estimado en la TAREA IV

Como la grafica indica, al inicio de la tarea, es necesario emplear varias horas de trabajo, ya que es primordial aprender a utilizar funciones de MATLAB para poder graficar los resultados. A medida que el tiempo transcurre, y gracias a la práctica obtenida el esfuerzo realizado se va acoplando al esfuerzo estimado. Por consiguiente se mantienen los plazos dentro del proyecto para esta tarea en específico.

TAREA V

Para finalizar el proyecto se realizara la presentación de los resultados en un reporte, el cual puede ser presentado en diferentes formatos, los más usuales, EXCEL de Microsoft y CSV el cual es un formato libre, y puede ser ejecutado en cualquier plataforma.

Las sub-tareas a realizar son pocas aunque con un nivel de complejidad aceptable, ya que se debe aprender a realizar la exportación de datos desde MATLAB a los distintos formatos.

Las sub-tareas a realizar son:

- Programar: Almacenar los datos de los resultados (código + pruebas unitarias)
- Programar: Exportar en formato EXCEL (código + pruebas unitarias)
- Programar: Exportar en formato CSV (código + pruebas unitarias)

Como muestra el grafico, la estimación para la tarea es exagerada, ya que el numero de sub-tareas es mínimo, sin embargo se distribuye el trabajo a través de los días para no saturar el desarrollo del proyecto.

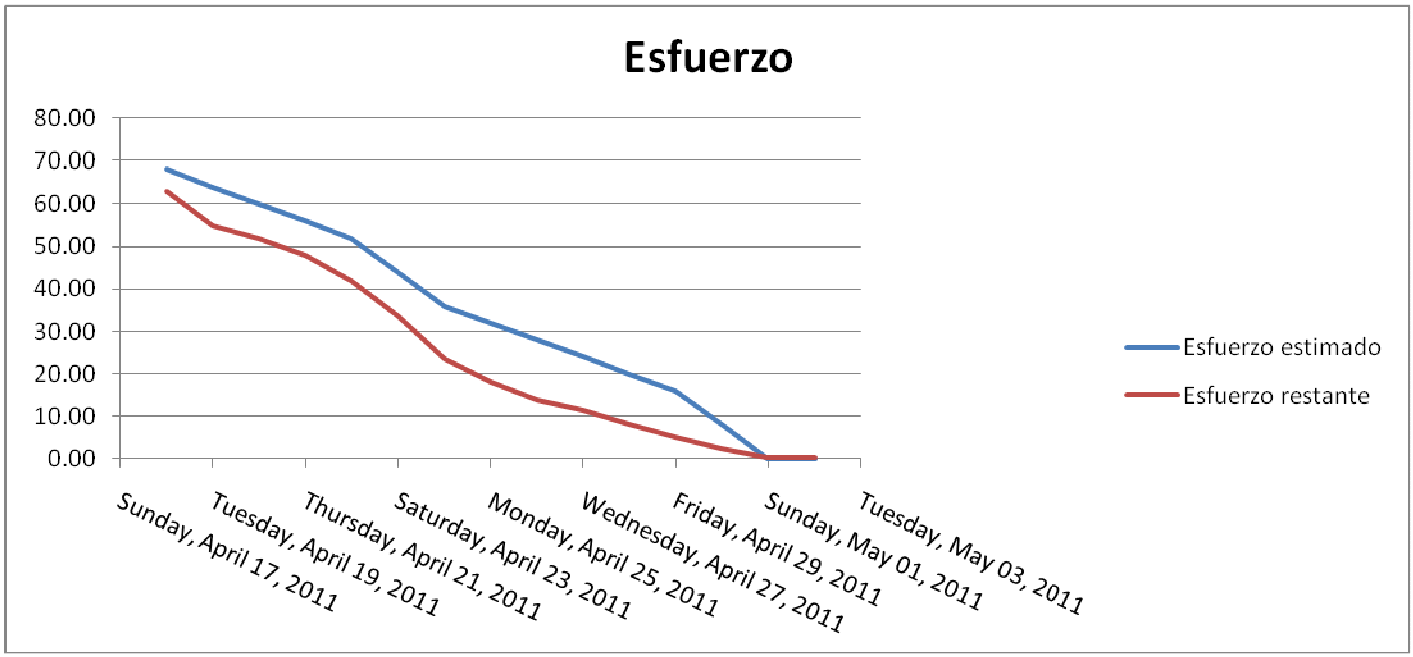


Figura 23 – Representación grafica del esfuerzo realizado y el esfuerzo estimado en la TAREA V

Como conclusión podemos mencionar, que la gestión del proyecto es una aproximación del trabajo real realizado, ya que explicar cada evento o suceso que se presento a lo largo del proyecto conllevaría explicaciones las cuales harían salir del tema principal el cual es el desarrollo del proyecto.

Según la estimación del tiempo que se tenía previsto, se puedo afrontar el proyecto de una manera ordenada, y secuencial.
Siguiendo la línea del esfuerzo estimado, el esfuerzo realizado fue variando pero sin ningún exabrupto, conteniendo el desarrollo del proyecto dentro de los límites establecidos.

Además, se pudo comprobar que la gestión del proyecto es un punto importante, ya que nos proporciona el medio para valorar el progreso que se tiene en un punto determinado del proyecto.

Para finalizar, decir que desarrollar un proyecto sin un plan de ejecución, es totalmente ineficiente, si bien es cierto que se puede llegar y cumplir un objetivo sin un plan previo, creo que profesionalmente hablando no importa llegar a un objetivo, sino saber que mientras se avanza hacia un objetivo se está aprovechando al máximo todos los recursos disponibles para el proyecto.

5 Pruebas

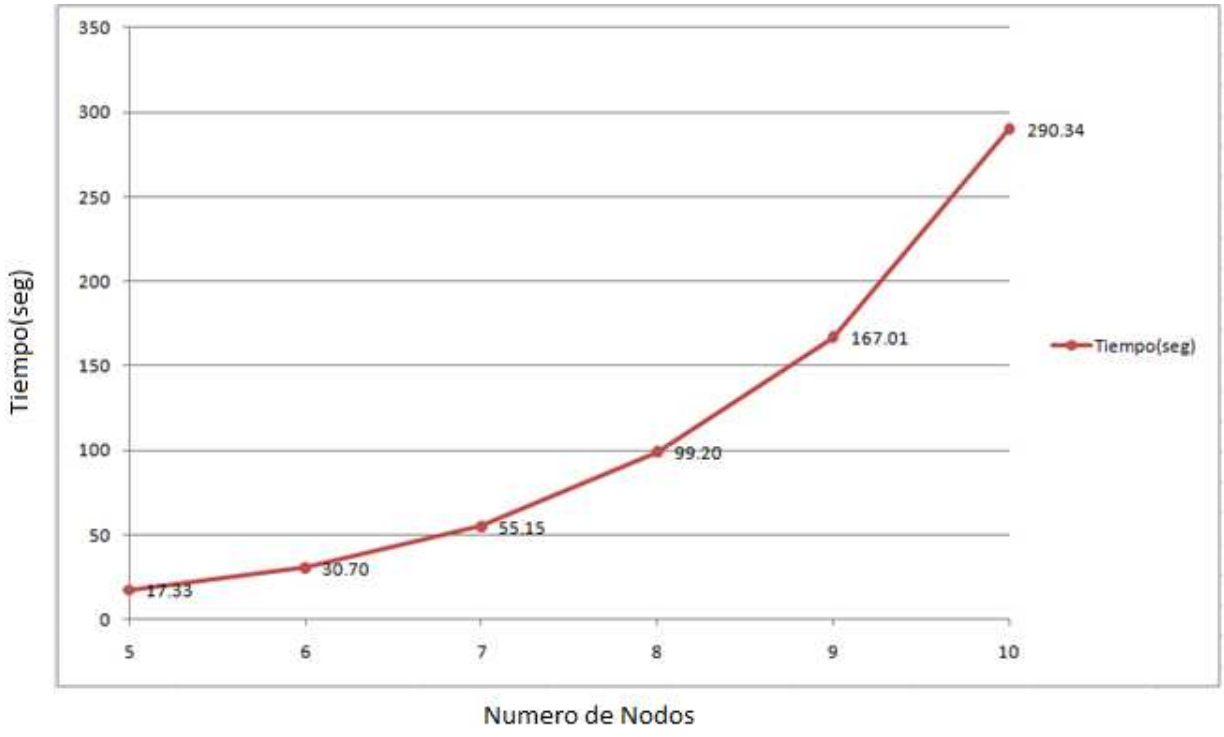
Para comprobar el tiempo de procesamiento del software, se realizan pruebas variando las opciones que configuran el comportamiento de los algoritmos genéticos.

Prueba 1

Determina el tiempo de procesamiento que tarda el software en encontrar una solución, variando el número de nodos Backbone existentes en la red.

La prueba es realizada con la siguiente configuración:

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Normal Dijkstra
- Mantener el Elitismo
- Selección probabilística
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- Población conformada por 15 cromosomas
- Numero de nodos Backbone (opción que se modifica)



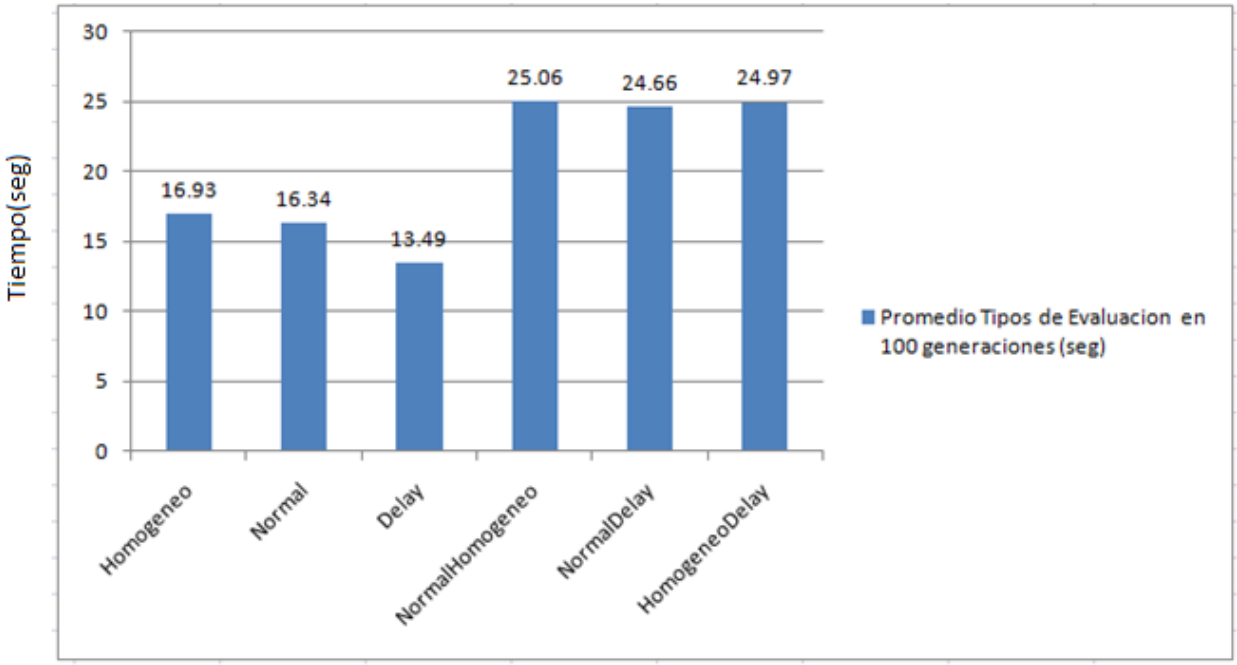
La grafica nos muestra como el tiempo de procesamiento es directamente proporcional al número de nodos Backbone existentes en la red, debido al aumento de datos que son evaluados al incrementar la cantidad de nodos Backbone en la red.

Prueba 2

Muestra el tiempo promedio de ejecución, existente entre los diferentes métodos de evaluación en el algoritmo genético.

La prueba es realizada con la siguiente configuración:

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Mantener el Elitismo
- Selección probabilística
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Tipo de evaluación (opción que se modifica)



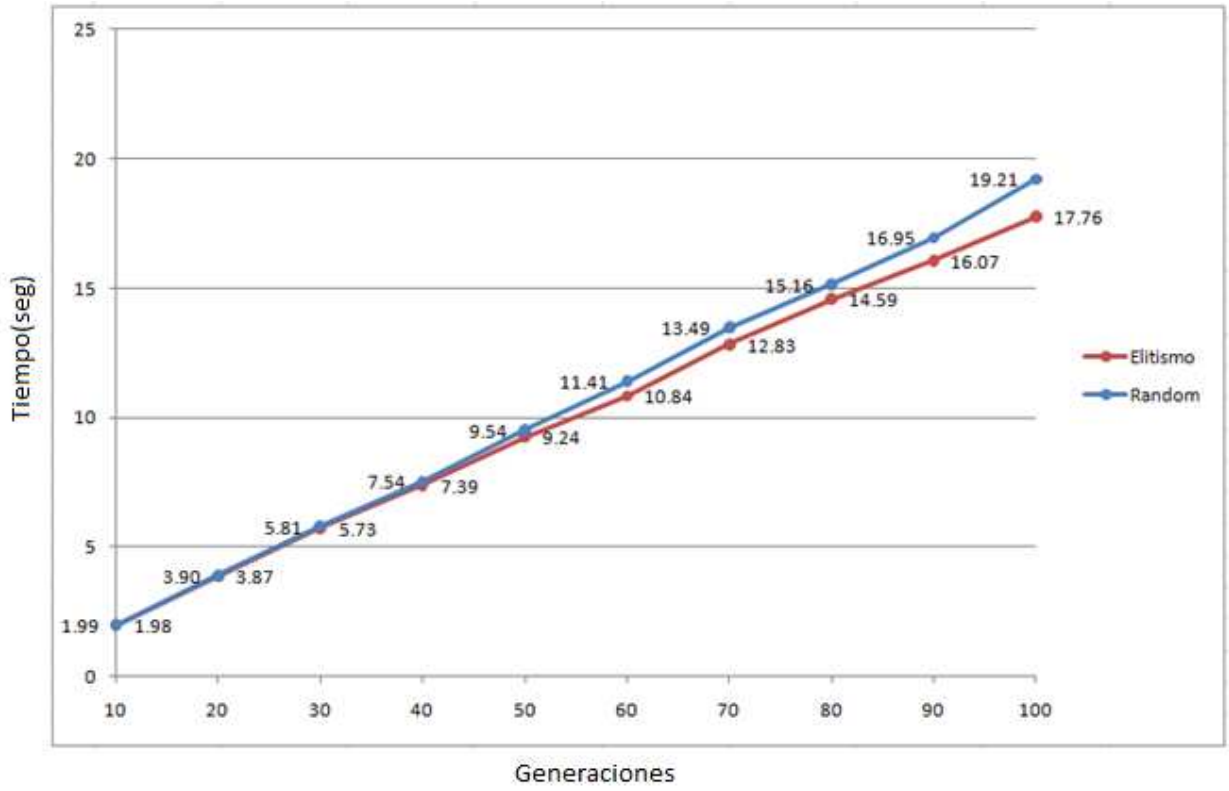
La grafica muestra como el tiempo de procesamiento se incrementa a medida que se toman en cuanta más de un tipo de evaluación.

Prueba 3

Muestra la influencia que genera en el tiempo de procesamiento, el escalado en las poblaciones de una generación, es decir, elegir el criterio de elitismo en la formación de las nuevas generaciones o realizar este proceso aleatoriamente.

La prueba es realizada con la siguiente configuración:

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Dijkstra Normal
- Selección probabilística
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Función de escalado (opción que se modifica)



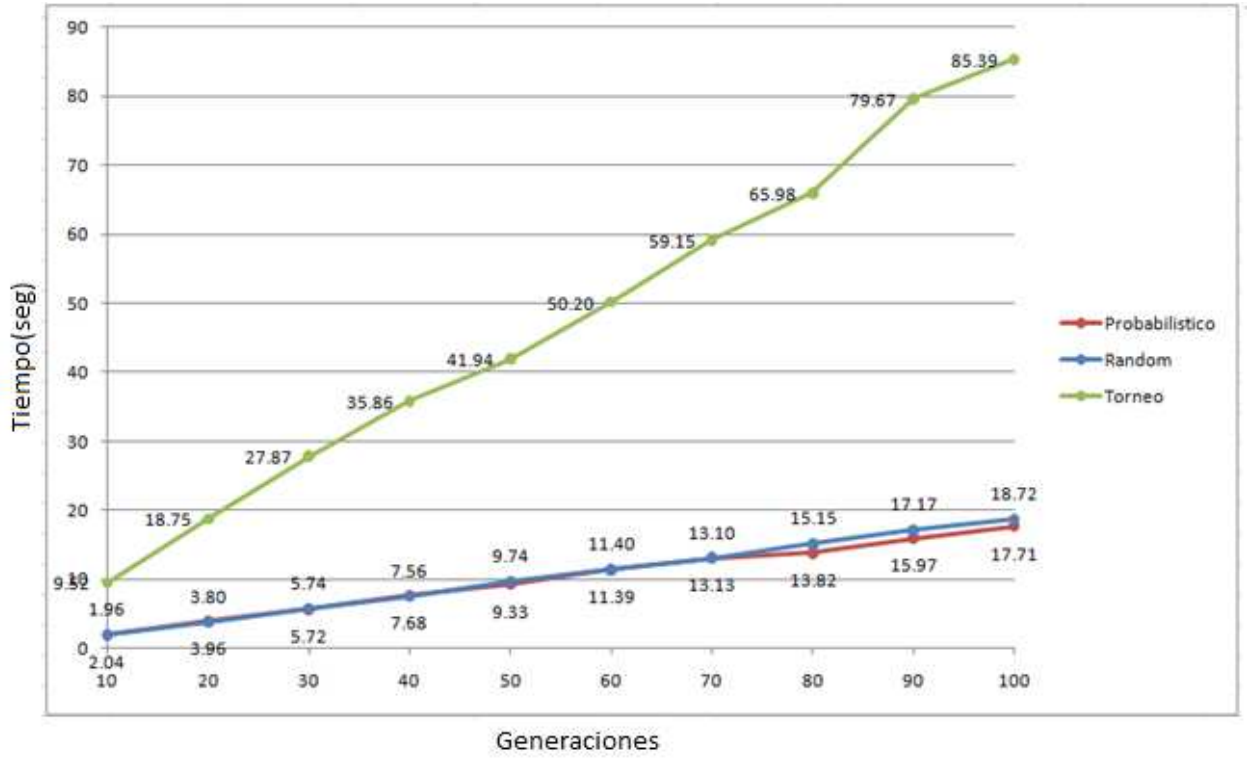
Los resultados demuestran que la variación de dicha opción no influye significativamente, ya que hasta un número de generaciones el valor que se obtiene es similar en ambos casos, puede ser posible un aumento de la influencia a partir del centenar de generaciones en el algoritmo genético.

Prueba 4

Determina el tiempo de procesamiento del algoritmo genético, al hacer variar el tipo de selección utilizada para elegir a los cromosomas en una generación.

La prueba es realizada con la siguiente configuración:

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Dijkstra Normal
- Mantener el Elitismo
- Si la selección es por torneo, el número de cromosomas por torneo es 5.
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Función de selección (opción que se modifica)



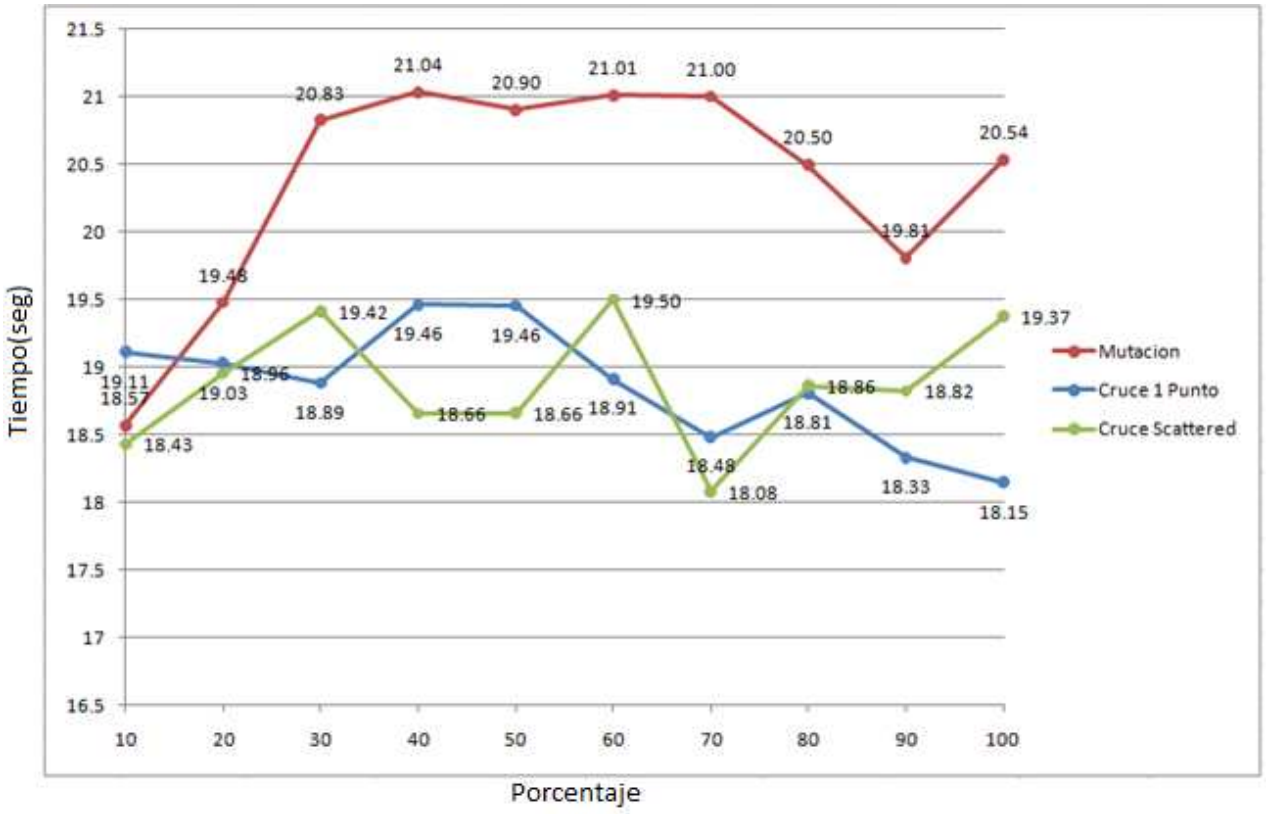
La grafica indica como la selección por torneo aumenta en gran medida el tiempo en el procesamiento del software debido a la evaluación previa que debe realizar para encontrar el mejor cromosoma de cada torneo que se forma para seleccionar los cromosomas en una generación.

Prueba 5

Determina la influencia que generan los métodos de reproducción, es decir métodos de cruce y mutación, según el porcentaje de utilización en el algoritmo genético.

La prueba es realizada con la siguiente configuración:

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Dijkstra Normal
- Mantener el Elitismo
- Selección probabilística
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Función de mutación (opción que se modifica)
- Función de cruce (opción que se modifica)



La grafica muestra una influencia mayor de la mutación frente al cruce en los algoritmos genéticos, debido a que en la mutación se busca nueva información para ser introducida en el algoritmo genético y en el cruce solo se realiza el intercambio de bloques constructivos.

6 Conclusiones

El desarrollo del PFC permite comprobar el comportamiento de un Algoritmo Genético aplicado a un problema real como es el diseño de redes, tener conocimiento de las ventajas y adversidades, para encontrar soluciones explorando un espacio de posibilidades, donde otras técnicas no tienen la capacidad de realizar.

Entre las ventajas se encuentra las múltiples propuestas de soluciones que un Algoritmo Genético puede brindar en una ejecución, debido a su aptitud para buscar soluciones. A diferencia del criterio humano, un Algoritmo Genético puede dilatar tanto los limites de un problema hasta encontrar soluciones extremas inimaginables para la lógica común.

Otra ventaja es la facilidad para representar el entorno de un problema dentro de un Algoritmo Genético, gracias a la carencia de datos necesarios para desarrollar un software.

La libertad que existe en los Algoritmos Genéticos, posibilita desarrollar aplicaciones para cualquier área del conocimiento, diferentes ámbitos de la investigación, incluso existe la posibilidad de indagar respuestas a situaciones extremas en las que las técnicas tradicionales no pueden alcanzar debido a su falta de flexibilidad y adaptabilidad al medio.

Entre las desventajas se encuentra el tiempo de procesamiento, porque en las pruebas realizadas se puede observar como aumenta este valor con el incremento de los nodos Backbone que se toman en cuenta.
Debido a que el tamaño del cromosoma se incrementa según el número de nodos Backbone existentes en la red, ya que se debe conformar un cromosoma que agrupe todas las posibles conexiones entre los nodos Backbone de la red. Lo que conlleva evaluar más datos incrementando el tiempo de procesamiento.

En conclusión, con la aplicación de la inteligencia artificial se consigue complementar el trabajo realizado por el personal capacitado, reduciendo el tiempo del proceso de diseño, obteniendo diseños óptimos, automatizando acciones en distintas situaciones, etc.

La aplicación de técnicas de Soft Computing conlleva una evolución en el área informática, porque las aplicaciones actuales no solo requieren satisfacer ciertos requisitos, también involucra brindar una mejor experiencia al usuario, encontrar soluciones a problemas que se presenten en tiempo real y adaptarse al medio según las acciones que ocurren.

En futuros trabajos es posible desarrollar aplicaciones capaces de modificar el diseño de una red existente, maximizando el rendimiento, reduciendo enlaces entre nodos, tener en cuenta el factor económico, etc. Según los requerimientos del usuario.

7 Bibliografía

[1] Holland, John. “Genetic algorithms” Scientific American, 1992, p. 66-72

[2] Sato, S., K. Otori, A. Takizawa, H. Sakai, Y. Ando y H. Kawamura. “Applying genetic algorithms to the optimum design of a concert hall.” Journal of Sound and Vibration, vol.258, no.3, 2002, p. 517-526

[3] Rizki, Mateen, Michael Zmuda y Louis Tamburino. ``Evolving pattern recognition systems." IEEE Transactions on Evolutionary Computation, vol.6, no.6, 2002, p. 594-609

[4] Zbigniew Michalewicz. “Genetic Algorithms + Data Structures = Evolution Programs” Third Edition Springer, 1992, p. 13-44

[5] Randy L. Haupt, Sue Ellen Haupt. “Practical Genetic Algorithms”, Second Edition Wiley – Interscience, 2004, p. 27-49

[6] Melanie Mitchell. “An Introduction to Genetic Algorithms”, First MIT Press paperback Edition, 1998, p. 1-44

[7] William M. Spears. “Evolutionary Algorithms The Role of Mutation and Recombination”, Springer, p. 3-18

[8] Marco Gestal, Daniel Rivero, Juan Ramon Rabuñal, Julian Dorado, Alejandro Pazos. “Introduccion a los Algoritmos Geneticos y la Programcion Genetica”, Universidad de la Coruña, 2010, p. 11-49

[9] Francisco Herrera Jose Luis Verdegay, Manuel Lozano. “Algoritmos Geneticos Apuntes del curso “Bioinformatica: Algoritmos Geneticos’’”, Universidad de Granada, 1993, p. 1-52

[10] James F. Kurose, Keith W. Ross. “Computer Networking A TOP-DOWN APPROACH”, Fifth Edition Pearson International Edition, 2010, p. 400-407

[11] Marcelo MEJÍA, David LÓPEZ. “Aplicación Práctica de Algoritmos Genéticos al Diseño de Redes”

[12] King-Tim Ko, Kit-Sang Tang. “Using Genetic Algorithms to Design Mesh Networks”

[13] Mitsuo GEN, Kenichi IDA, Jongryul KIM. “A Spanning Tree-Based Genetic Algorithm for Bicriteria Topological Network Design”

[14] Jong Ryul Kim, Mitsuo Gen. “Genetic Algorithm for Solving Bicriteria Network Topology Design Problem”

[15] El-Sayed M. El-Alfy. “MPLS Network Topology Design Using Genetic Algorithms”

[16] Bassam Al-Bassam, Abdulmohsen Alheraish, Saad Haj Bakry. “A tutorial on using genetic algorithms for the design of network topology”

[17] Diego Orlando Barragan Guerrero. “Manual de interfaz grafica de usuario en matlab”

[18] José Javier Astrain. “Material de estudio de Reconocimiento de Patrones”

[19] Raul Orduna. “Material de estudio de Laboratorio de Inteligencia Artificial”

8 Referencia Bibliográfica

[20] https://www.msu.edu/~perryma3/cas492/att_backbone_large.gif

[21] http://es.wikipedia.org/wiki/Problema_del_viajante

[22] http://foro.colombiaunderground.org/index.php?action=printpage;topic=5047.0

[23] http://www.slideboom.com/presentations/88746/Reynoso---Algoritmo-genetico

[24] http://www.heatonresearch.com/articles/65/page1.html

[25] http://www.obitko.com/tutorials/genetic-algorithms/example-function-minimum.php

[26] http://www.obitko.com/tutorials/genetic-algorithms/tsp-example.php

[27] http://www.rennard.org/alife/english/gavgb.html

[28] http://www.palomatica.info/juckar/googlemap/

[29] http://es.wikipedia.org/wiki/Desviaci%C3%B3n_est%C3%A1ndar

[30] http://www.disfrutalasmatematicas.com/datos/desviacion-estandar.html

[31] http://www.spssfree.com/spss/analisis2.html

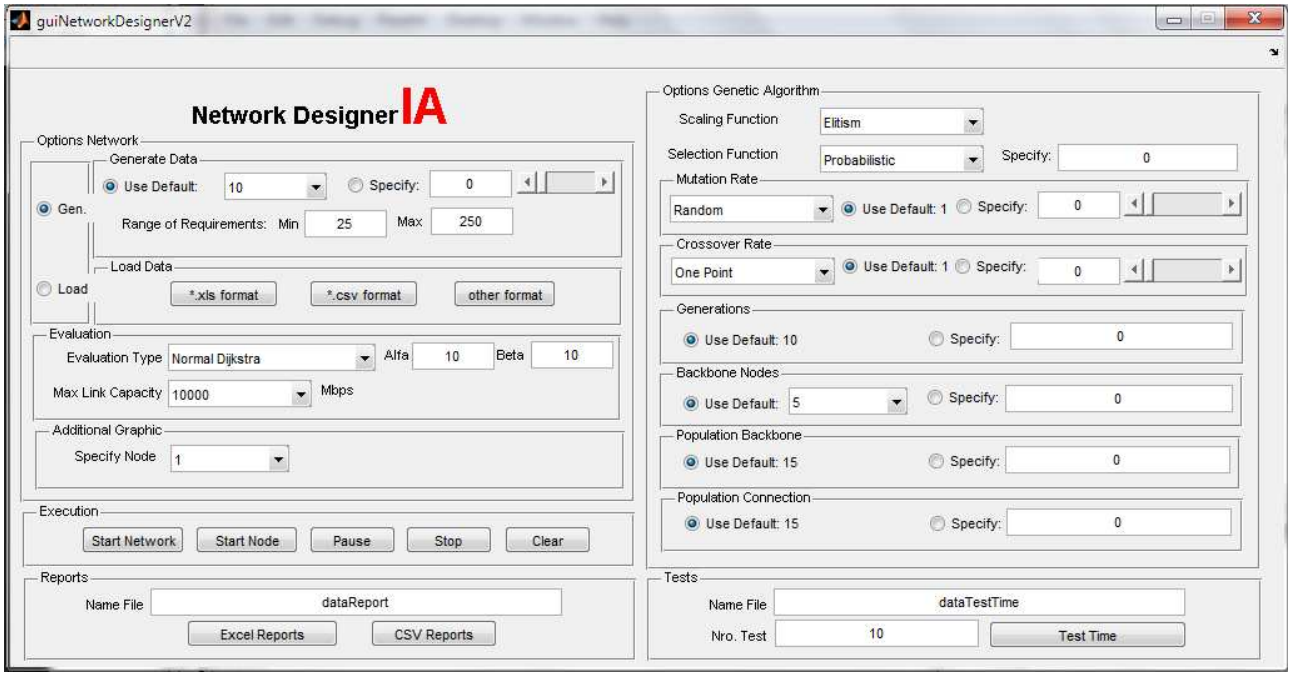
[32] http://es.wikipedia.org/wiki/Gravedad

[33] http://es.wikipedia.org/wiki/Ley_de_gravitaci%C3%B3n_universal

[34]http://www.economia48.com/spa/d/gravitacion-del-comercio-al-por-menor/gravitacion-del-comercio-al-por-menor.htm

9 Anexo I
Ejemplo del funcionamiento del Software

Al iniciar el software se visualiza la interfaz grafica principal.

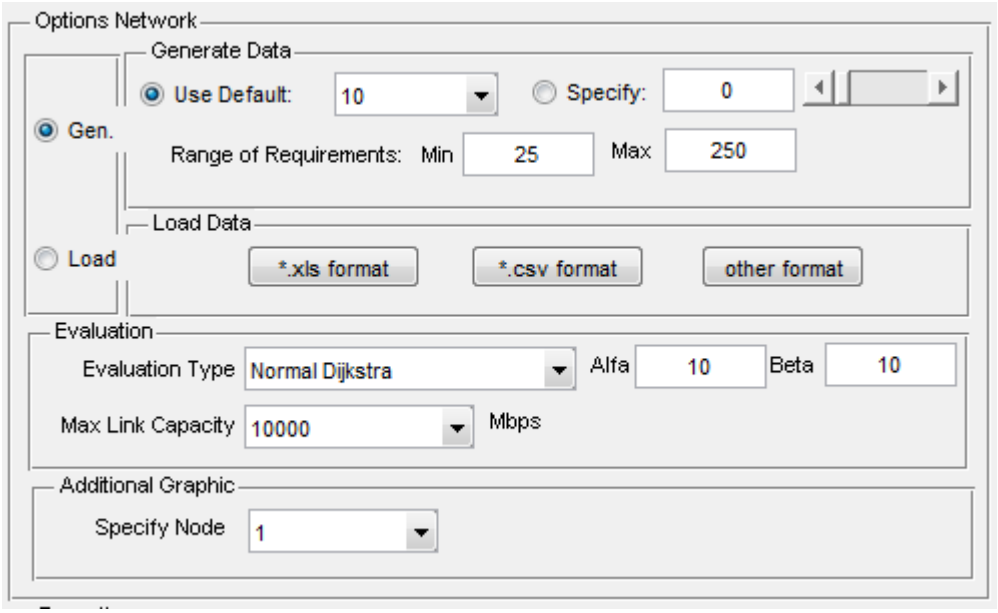


La interfaz grafica está dividida en las siguientes regiones:

- Opciones de Red
- Opciones del Algoritmo Genético
- Opciones de Ejecución
- Opciones de Reportes
- Opciones de Pruebas

A continuación se procede a describir en detalle cada una de los componentes de la interfaz grafica.

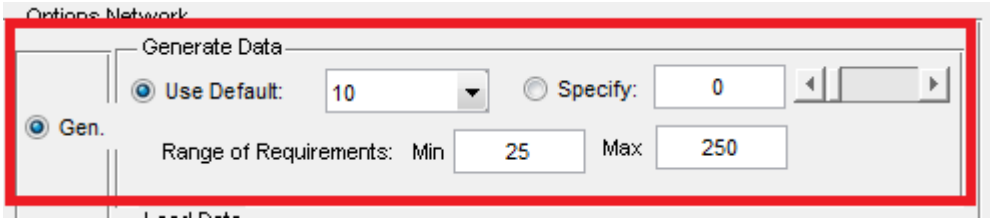
Opciones de Red



La primera región contiene las opciones correspondientes a la red, dividida en las siguientes áreas:

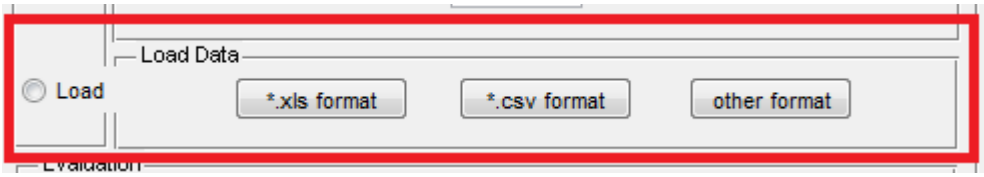
- área de generación o carga de datos iniciales
- opciones de evaluación
- opciones del grafico adicional

En el área de generación de datos, es posible seleccionar el número de nodos que existirá en la red, y el rango de los requerimientos que existe entre los nodos, es decir el ancho de banda que un par de nodos necesita.

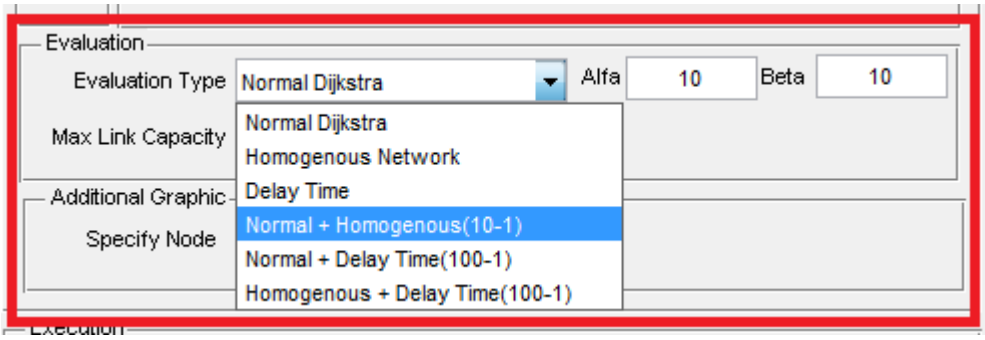


En el área de carga de datos es posible seleccionar un archivo con un formato específico para que el software pueda leer la información que dicho archivo presenta.

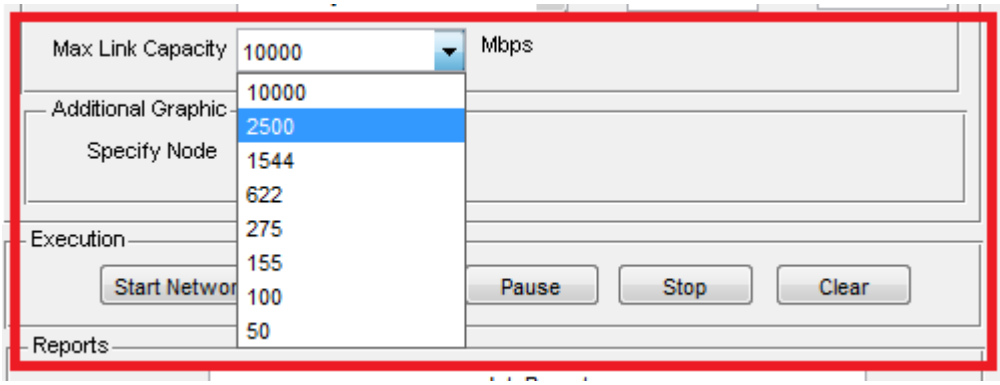
En el formato Excel, se selecciona un archivo.
En el formato CSV, se da el caso de que los datos están almacenados individualmente, es por ello que se selecciona el archivo denominado con el nombre del proyecto, de tal manera que el software pueda leer los datos de los demás archivos, simplemente adicionando el nombre del dato al nombre del archivo que se selecciono.



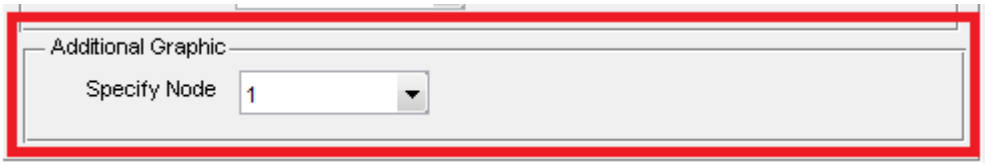
En el área de evaluación, se selecciona el tipo de evaluación que utilizara el algoritmo genético encargado de optimizar el costo de los enlaces en la red.
Se implemento una variedad de métodos de evaluación, métodos que tomen en cuenta solo un parámetro, como aquellos que combinen los parámetros para realizar la evaluación.
En los métodos que utilizan más de un parámetro, es necesario definir los valores Alfa y Beta, los cuales controlan la importancia que un parámetro puede tener frente al otro.



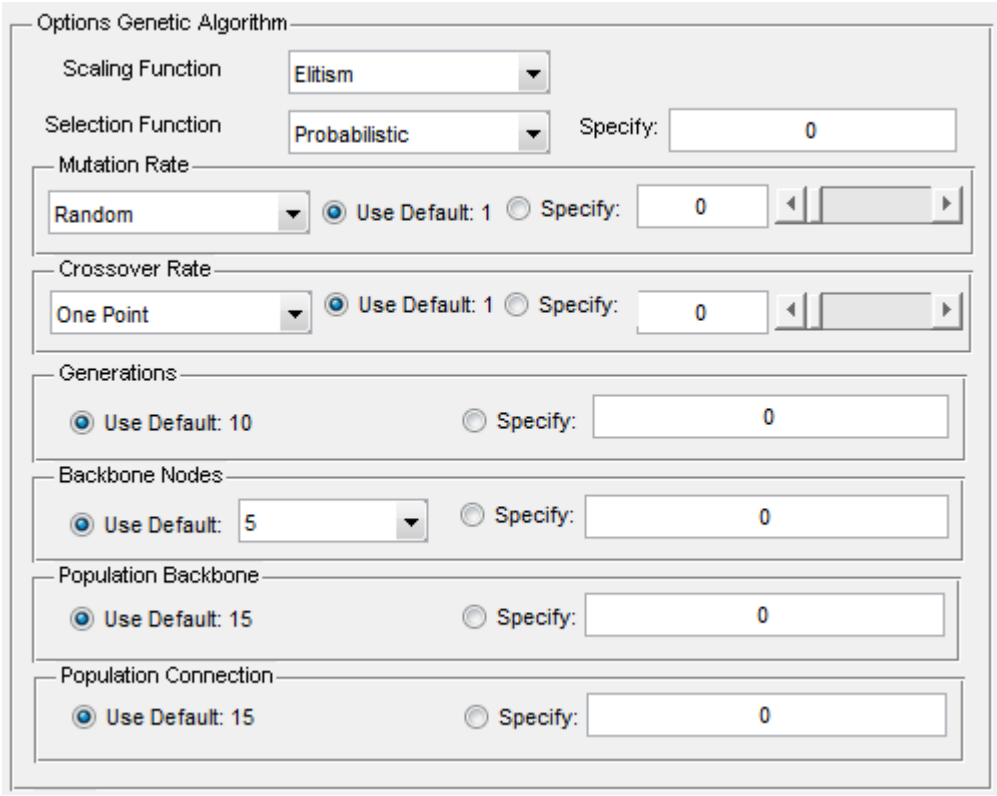
También se puede elegir la capacidad máxima de los enlaces disponibles, es decir la capacidad límite que un enlace puede presentar en la red, de tal manera que el software solo tomara en cuenta los enlaces existentes entre el primer enlace y el límite establecido.



En el área de las opciones del grafico adicional, se puede seleccionar el nodo para el cual se genere un grafico solo con la información referente a dicho nodo.



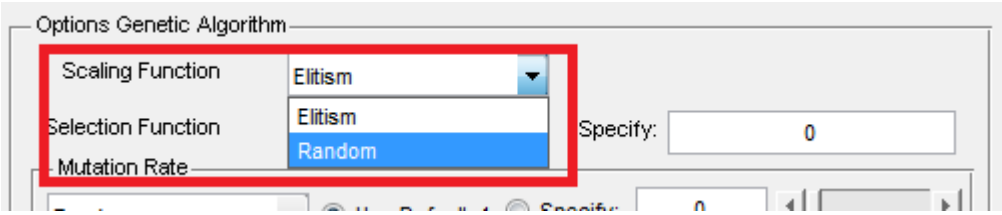
Opciones del Algoritmo Genético



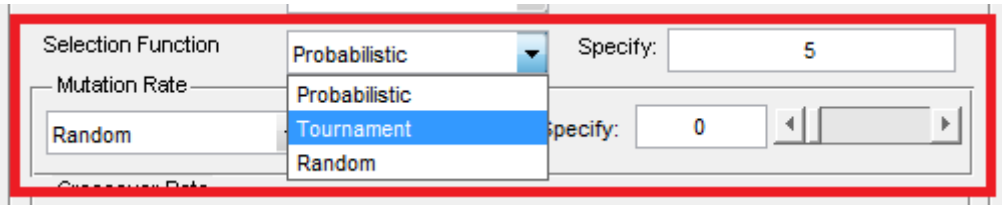
En esta sección de la interfaz grafica se seleccionan las opciones necesarias para el funcionamiento del algoritmo genético encargado de formar el núcleo Backbone de la red.

La sección contiene varias opciones las cuales serán descritas a continuación:

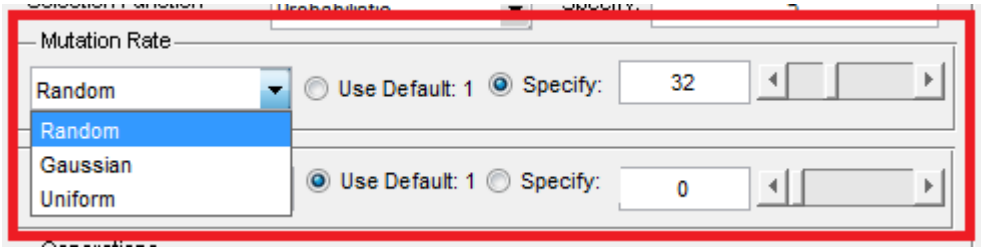
En la opción “Scaling Function” podemos determinar si el algoritmo genético mantendrá el elitismo al momento de generar las nuevas poblaciones de cromosomas, de no seleccionarse el elitismo el procedimiento será aleatorio.



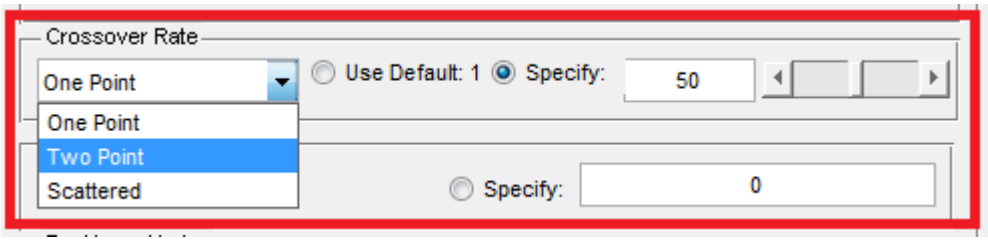
En la opción “Selection Function” podemos determinar tres tipos de métodos de selección, en donde si elegimos la opción “Tournament” debemos indicar en número de cromosomas por grupo que se formara para determinar el mejor cromosoma del grupo.



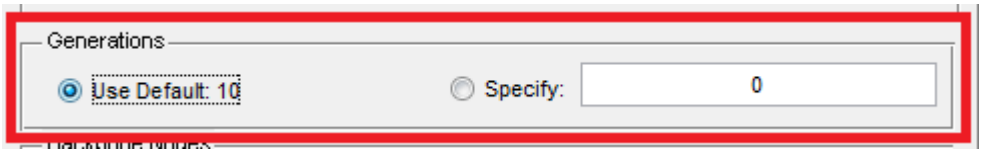
En la opción “Mutation Rate” podemos determinar el tipo de mutación, además del porcentaje de mutación que se requiere.
El porcentaje puede ser el por defecto o se puede especificar un valor porcentual diferente.



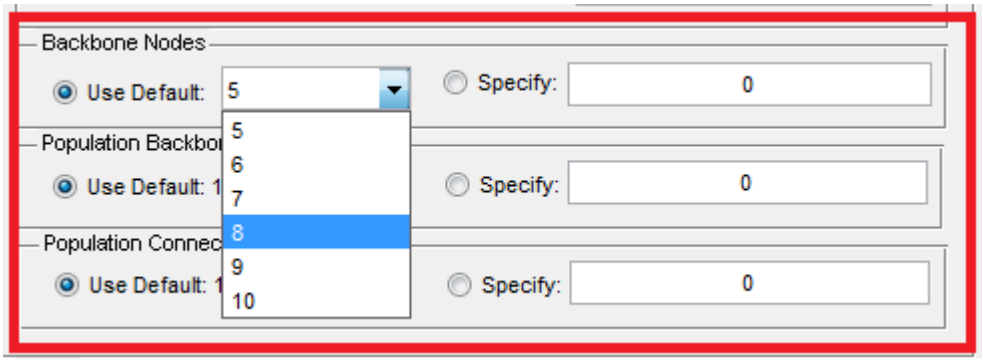
En la opción “Crossover Rate” podemos determinar el tipo de cruce, además del porcentaje de cruce que se requiere.
El porcentaje puede ser el por defecto o se puede especificar un valor porcentual diferente.



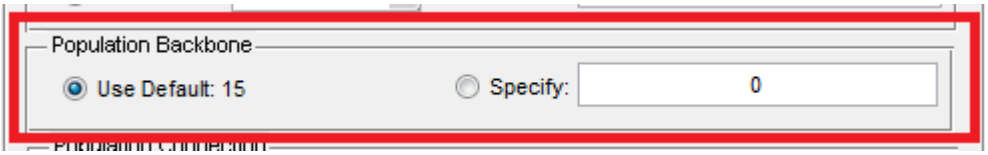
En la opción “Generations” podemos determinar el número de iteraciones que el algoritmo genético tomara para realizar las iteraciones correspondientes.
El numero de generaciones puede ser el por defecto o se puede especificar un valor diferente.



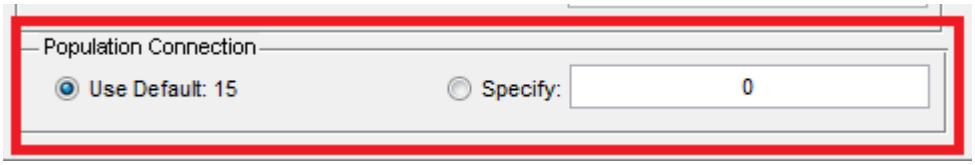
En la opción “Backbone Nodes” podemos determinar el número de nodos backbone que la red requiere.
El número de nodos backbone puede ser elegido de los más usuales que están por defecto o se puede especificar un valor diferente.



En la opción “Population Backbone” podemos determinar el número de cromosomas que tendrá la población de una generación correspondiente al primer algoritmo genético.
El número de la población puede ser el por defecto o se puede especificar un valor diferente.



En la opción “Population Connection” podemos determinar el número de cromosomas que tendrá la población de una generación correspondiente al segundo algoritmo genético.
El número de la población puede ser el por defecto o se puede especificar un valor diferente.

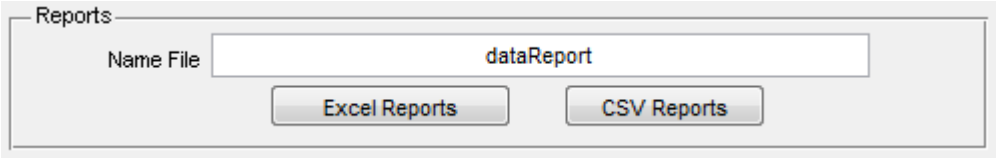


Opciones de Ejecución

Cuando presionamos el botón “Start Network” el software empieza su ejecución con los valores determinados por el usuario, devolviendo como repuesta el diseño de red.

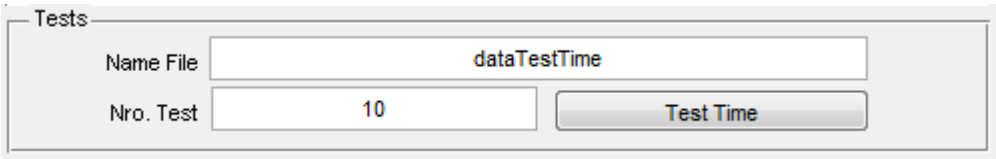
Al presionar el botón “Star Node” generamos un grafico para un nodo específico, con la información detallada correspondiente a dicho nodo.

Opciones de Reportes



Presionando el botón correspondiente al formato, se genera el mismo con el nombre establecido por el usuario en el campo habilitado.

Opciones de Pruebas



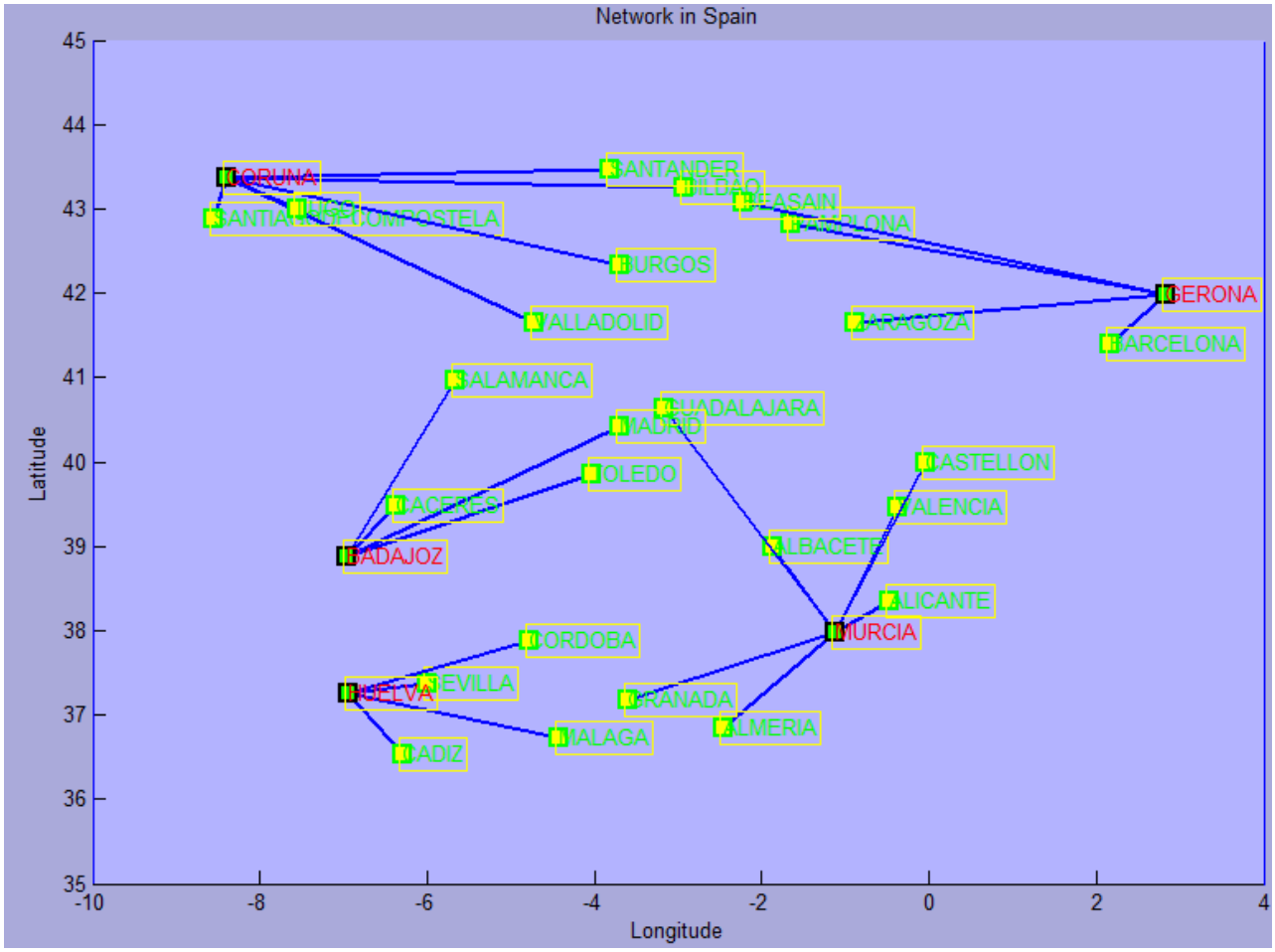
En la sección de pruebas, se puede ejecutar el software un número determinado de veces, con el fin de obtener información referente al tiempo que dura el cálculo de una solución en cada ejecución.
Dichos datos son exportados en un archivo denominado por el usuario en el campo habilitado.

Una vez seleccionadas las opciones, procedemos a ejecutar el algoritmo para obtener una solución al problema descrito por las opciones elegidas.

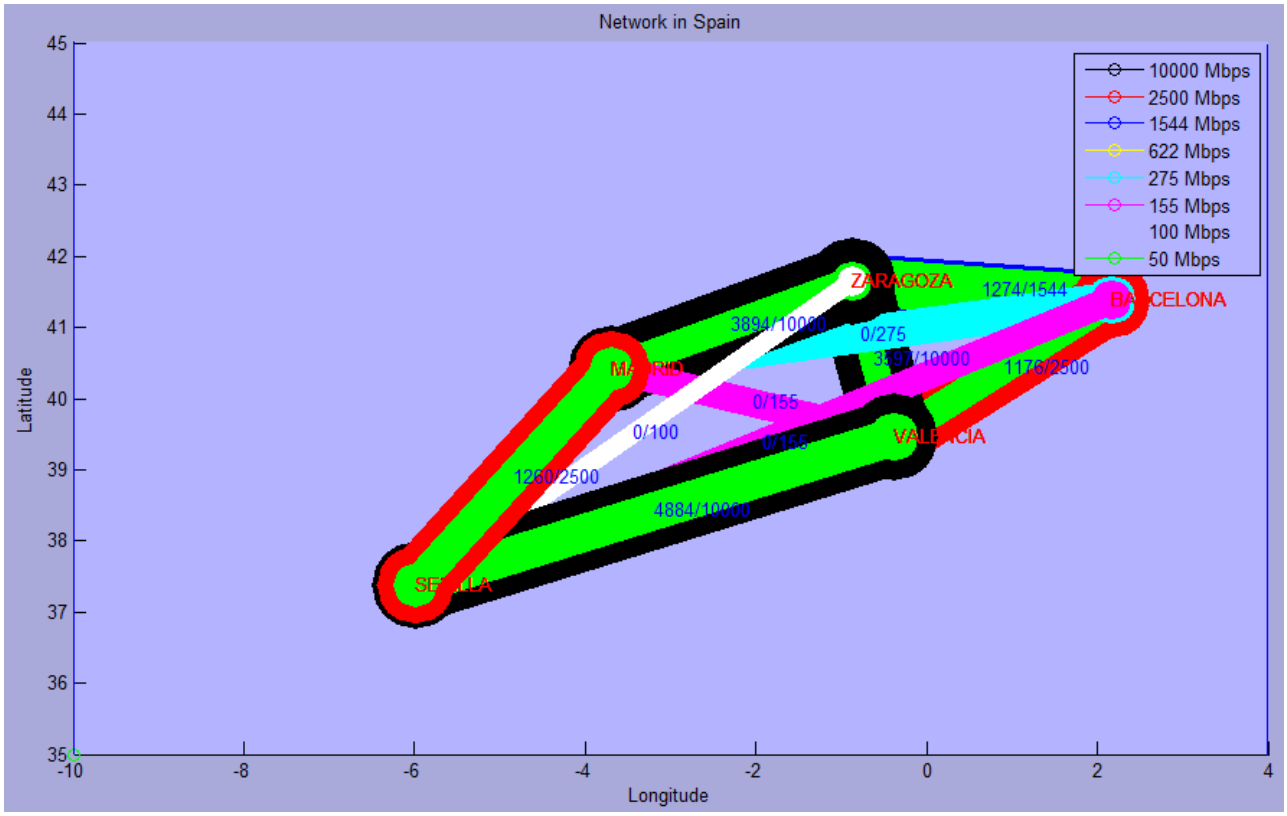
En el grafico se muestra la solución del algoritmo genético encargado de formar el núcleo Backbone.
Los nodos Backbone son resaltados, y se puede observar como los demás nodos, se acoplan al nodo Backbone más cercano, formando de esta manera grupos simétricos.

La representación de los nodos en el grafico, es mediante el uso de un plano cartesiano donde los ejes abscisa y ordenada representan la longitud y latitud respectivamente.

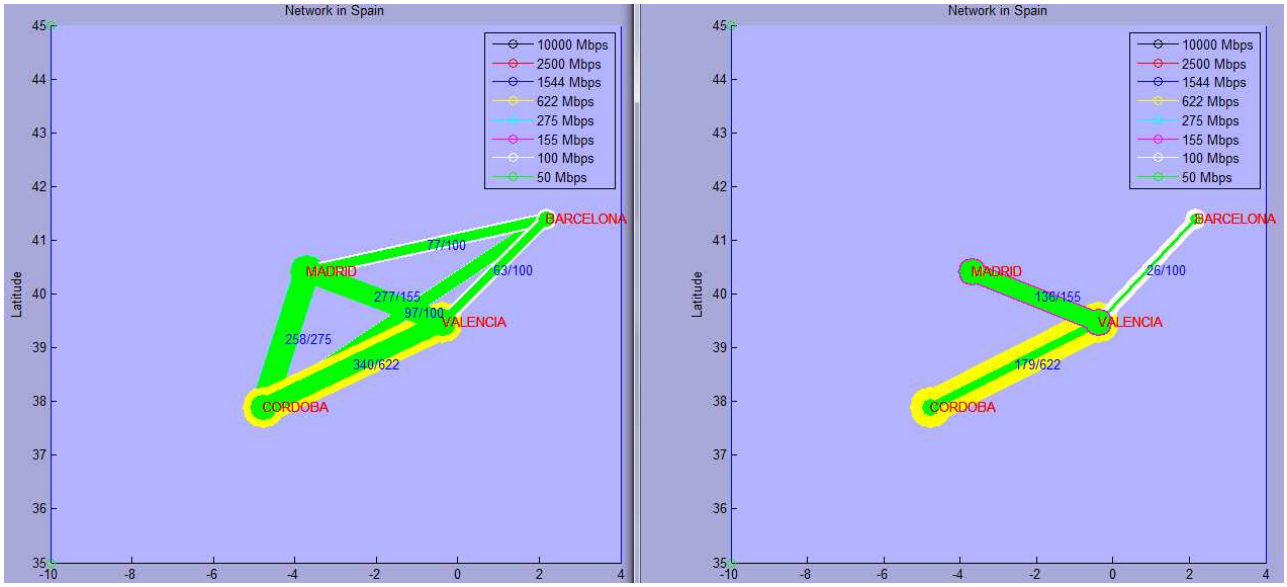
Cada nodo queda representado en su posición real según su latitud y longitud.



El siguiente grafico muestra los nodos Backbone conectados mediante los enlaces necesarios para satisfacer los requerimientos de la red.
Se puede observar la representación de los enlaces asignando un determinado color según la capacidad del mismo, los requerimientos en cada enlace y su capacidad correspondiente.



La imagen muestra a la izquierda el grafico completo del diseño de una red y a la derecha el grafico correspondiente a un nodo en específico.
El grafico para un nodo específico proporciona información importante, porque podemos conocer cuánto ancho de banda necesita el nodo para cumplir con los requisitos, el impacto que el nodo presenta en la red, etc.



Resultados en formato texto

Así mismo el software brinda información en formato texto, como por ejemplo:

Los requerimientos de ancho de banda que existe entre los nodos

0	22	17	24	6	17	11	22	9	14	13	22	19	9	11
8	0	18	16	8	8	15	23	16	6	6	21	14	13	21
12	16	0	19	22	12	18	24	7	17	13	11	10	20	25
9	21	9	0	25	21	13	20	15	21	12	6	17	23	9
14	20	6	24	0	20	16	9	15	15	25	22	24	19	13
24	15	10	13	19	0	16	20	25	24	16	24	7	6	11
17	16	23	16	14	16	0	19	5	21	8	15	10	12	18
8	11	9	9	12	23	14	0	13	9	24	13	22	17	13
23	21	14	21	23	14	12	17	0	23	19	13	20	24	16
16	11	6	9	7	14	5	23	18	0	5	6	9	14	8
5	20	12	21	14	14	6	6	7	17	0	10	22	22	24
15	9	10	16	20	12	14	18	23	8	19	0	17	14	23
13	9	18	17	12	21	25	25	8	10	5	17	0	7	13
23	16	12	9	14	24	7	14	22	6	19	25	11	0	8
19	23	17	23	9	20	12	13	8	21	17	20	21	6	0

Los datos de las coordenadas de cada nodo

1.0000	40.4167	-3.7003
2.0000	41.3879	2.1698
3.0000	42.8180	-1.6442
4.0000	42.8804	-8.5463
5.0000	41.6563	-0.8765
6.0000	36.7196	-4.4200
7.0000	37.3826	-5.9963
8.0000	37.8847	-4.7792
9.0000	37.9834	-1.1299
10.0000	39.4702	-0.3768
11.0000	40.9650	-5.6630
12.0000	43.2570	-2.9234
13.0000	41.6529	-4.7284
14.0000	38.8786	-6.9703
15.0000	38.9977	-1.8601

El nombre de las ciudades a las que corresponde

```
'MADRID'
'BARCELONA'
'PAMPLONA'
'SANTIAGODECOMPOSTELA'
'ZARAGOZA'
'MALAGA'
'SEVILLA'
'CORDOBA'
'MURCIA'
'VALENCIA'
'SALAMANCA'
'BILBAO'
'VALLADOLID'
'BADAJOZ'
'ALBACETE'
'ALMERIA'
'GRANADA'
'HUELVA'
'CADIZ'
```

La matriz distancia que se genera calculando la distancia entre todos los nodos.

```
  0    0    0    0    0    0    0    0    0    0    0    0    0
505    0    0    0    0    0    0    0    0    0    0    0    0
317  353    0    0    0    0    0    0    0    0    0    0    0
487  899  563    0    0    0    0    0    0    0    0    0    0
274  256  144  646    0    0    0    0    0    0    0    0    0
416  771  719  771  629    0    0    0    0    0    0    0    0
392  832  709  649  648  158    0    0    0    0    0    0    0
297  712  610  641  536  134  121    0    0    0    0    0    0
350  473  540  831  409  323  434  321    0    0    0    0    0
303  304  387  782  247  468  542  421  178    0    0    0    0
177  658  392  320  407  485  400  351  512  479    0    0    0
323  468  115  459  245  739  704  618  606  472  341    0    0
162  576  285  343  320  550  488  419  511  441  109  232    0
328  826  627  465  603  329  187  221  519  573  258  594  363
223  434  426  709  307  339  404  283  130  138  391  482  383
```

La evolución de la población de cromosomas al aplicar los algoritmos genéticos, con el respectivo valor de adaptación del mejor cromosoma.

6	27	8	2	30
29	25	7	6	11
19	29	15	10	2
4	27	6	23	22
7	13	26	25	28
16	9	20	19	13
6	17	1	30	24
28	17	10	9	13
12	30	6	27	11
27	16	30	5	22
2	11	17	7	23
16	25	6	13	7
15	16	27	18	29
23	3	6	12	21
5	20	22	12	10
27	13	23	11	28
4	12	18	6	30
13	30	20	15	9
6	19	7	15	24
21	15	10	4	20
17	25	20	1	19
14	10	27	17	23
18	6	10	14	4
22	25	30	5	23

```
minimoValor =

      864279


mCromosomasRedes =

    16    12     2    24    14
    16     9    20    19    13
    27     3     2    24    14
     1    16    27     5    22
    16    12    17    24    14
    16     9    20     6    13
    16    30    20     6     9
     1     9    20    19    26
    19    30    20    15     9
     1     9    20    19    14
    15     1     2    24    14
     1    16    27     2    29
    19    29    18    10     2
    21    29    15    10     2
    28     9    20    19    13
    16    10    20    19    13
    16    26    15    12    30
    29    12     2    24    14
    29     9    20     1    19
    15    25     7     6    11
    12    25    19     6    11
    29    30     6    23    11
```

```
minimoValor =

      850503


mCromosomasRedes =

    19    20    22     2    10
    16    12     2    24    14
    27    30     6    15     9
    19    27     2    24    14
    19    11    27     5    22
     1     6    22     2    10
    16    10     1    19    13
    16    10     2    24    14
    16    28    19     6    11
     1    23     2    24    14
    12    20    22     4    10
     1    25    18     6    11
    16    29    15    12     2
    21    10    20    23    13
    19     1    10     9    13
    16    20    29     2    10
     1     9    25    26    13
    16     9    20     2    26
    24    16    18     5    22
     1    29    15    26     2
     1     3    15    29     2
     3     1     4    24    14
```

```
minimoValor =

      848494


mCromosomasRedes =

    19    20    11     2    10
    19    20    22     2    10
    16    10    20    15    13
    21    19     5    26    13
     1    28    19    17    21
     1    20    17     2    10
     9    16    18     1    26
    28    16    18     5    17
    16     6    20    23    13
    21     9    24    26    13
    16    14    20    23    13
    10    27     5    24    14
     2    10     1    25    14
    21     3    20    23    13
    21     1    18    16    22
    28     1    20    23    13
    19    10     1     2    14
     1    21    11     2    10
     2    16    23     1    22
    28    19    20    23    26
```

De esta manera podemos observar la evolución de la población y la convergencia de la solución hacia la más optima hasta llegar al límite de iteraciones.

Una vez completadas las iteraciones, podemos observar la matriz de grupos que se forma con el mejor cromosoma de nodos Backbone.

```
matrizGruposRedes =

  Columns 1 through 19

    19     6     7     8    16    17    18     0     0     0     0     0
    20     1    29     0     0     0     0     0     0     0     0     0
    11     4    12    13    14    22    24    26    27    28    30     0
     2     3    23     0     0     0     0     0     0     0     0     0
    10     5     9    15    21    25     0     0     0     0     0     0
```

Los resultados correspondientes al segundo algoritmo genético pueden ser los siguientes:

La nueva matriz de requerimientos generada a partir de los grupos formados por el primer algoritmo genético.
Todos los nodos de un grupo, tienen como único acceso a la red el nodo Backbone, lo cual implica formar una nueva matriz de requerimientos solo entre los nodos Backbone pero garantizando el ancho de banda que los demás nodos miembros de un grupo necesitan.

0	149	174	55	168
113	0	185	51	190
175	189	0	58	237
43	28	59	0	63
191	187	273	66	0

La población existente en una generación del algoritmo genético, se puede observar como están representados los enlaces por un valor según su capacidad.

6	3	8	0	7	5	4	5	6	1
7	0	2	1	7	1	5	5	5	7
0	7	4	6	2	4	6	8	4	1
0	5	6	6	3	8	4	8	1	2
6	7	6	2	1	7	6	2	2	3
7	7	5	5	2	6	6	4	4	4
2	5	7	7	6	2	6	1	1	2
1	5	1	7	6	7	4	5	7	4
2	8	0	3	8	3	2	1	7	1
3	7	4	5	5	6	0	4	0	7
3	7	2	5	8	0	0	0	5	5
1	6	5	1	1	1	6	1	2	2
1	7	6	6	5	4	3	5	1	1
0	8	8	1	4	5	2	6	4	3
2	6	7	6	3	8	2	4	8	2

El vector de requerimientos que la red genera con los enlaces que el algoritmo genético propone como solución.

0	0	0	392	158	0	0	754	2040	516

El vector que contiene la representación de los enlaces que el algoritmo genético propone como solución.

0	0	6	2	3	7	6	2	2	3

El tiempo que tarda el algoritmo genético en encontrar la solución, representado en segundos.

5.5639

Proyecto de Fin de Carrera

Optimización del costo de enlaces en una red

2011

Victor Sanjinez

INDICE

1 OPTIMIZACION

2 DISENO DE REDES

3 PROBLEMA

4 SOLUCION

5 ESTRUCTURA DE UN ALGORITMO GENETICO

6 DETALLES DE LA SOLUCION

7 PRIMER ALGORITMO GENETICO

INDICE

8 SEGUNDO ALGORITMO GENETICO

9 COMPLEMENTOS DEL SOFTWARE

10 PRUEBAS

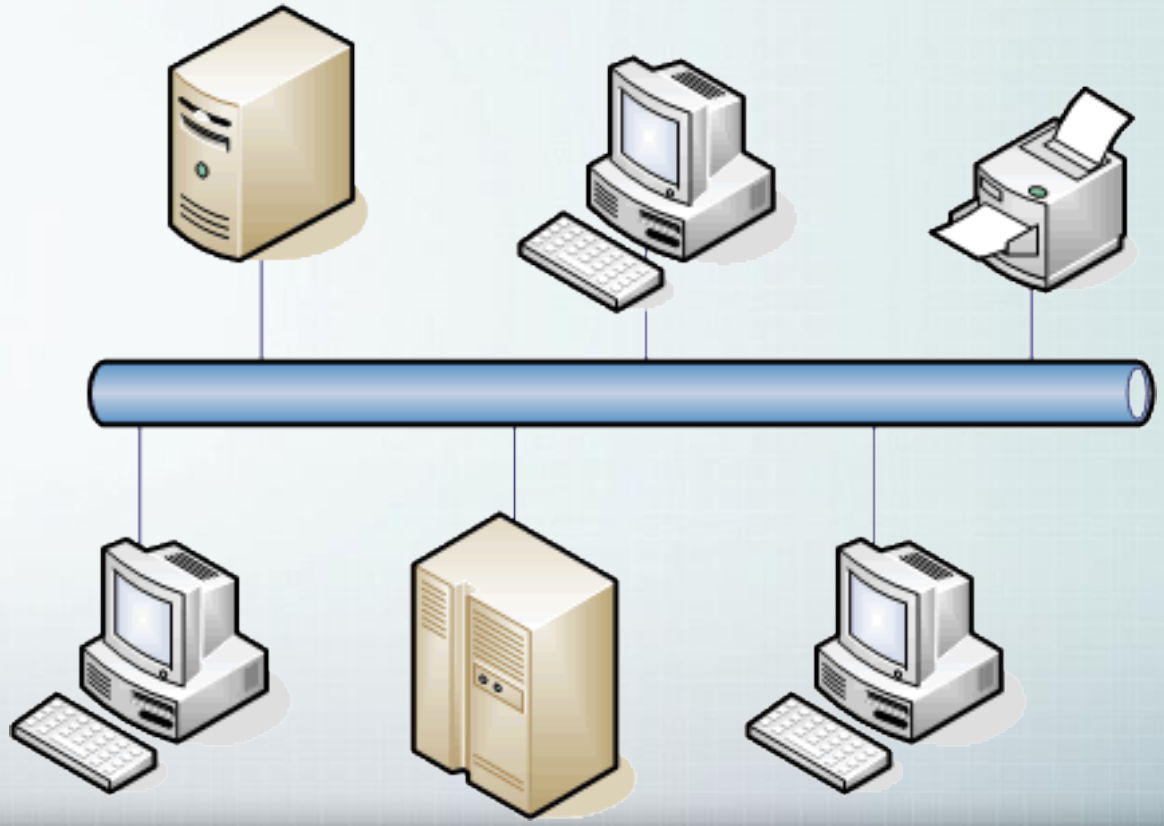
4 CONCLUSION

5 TRABAJOS FUTUROS

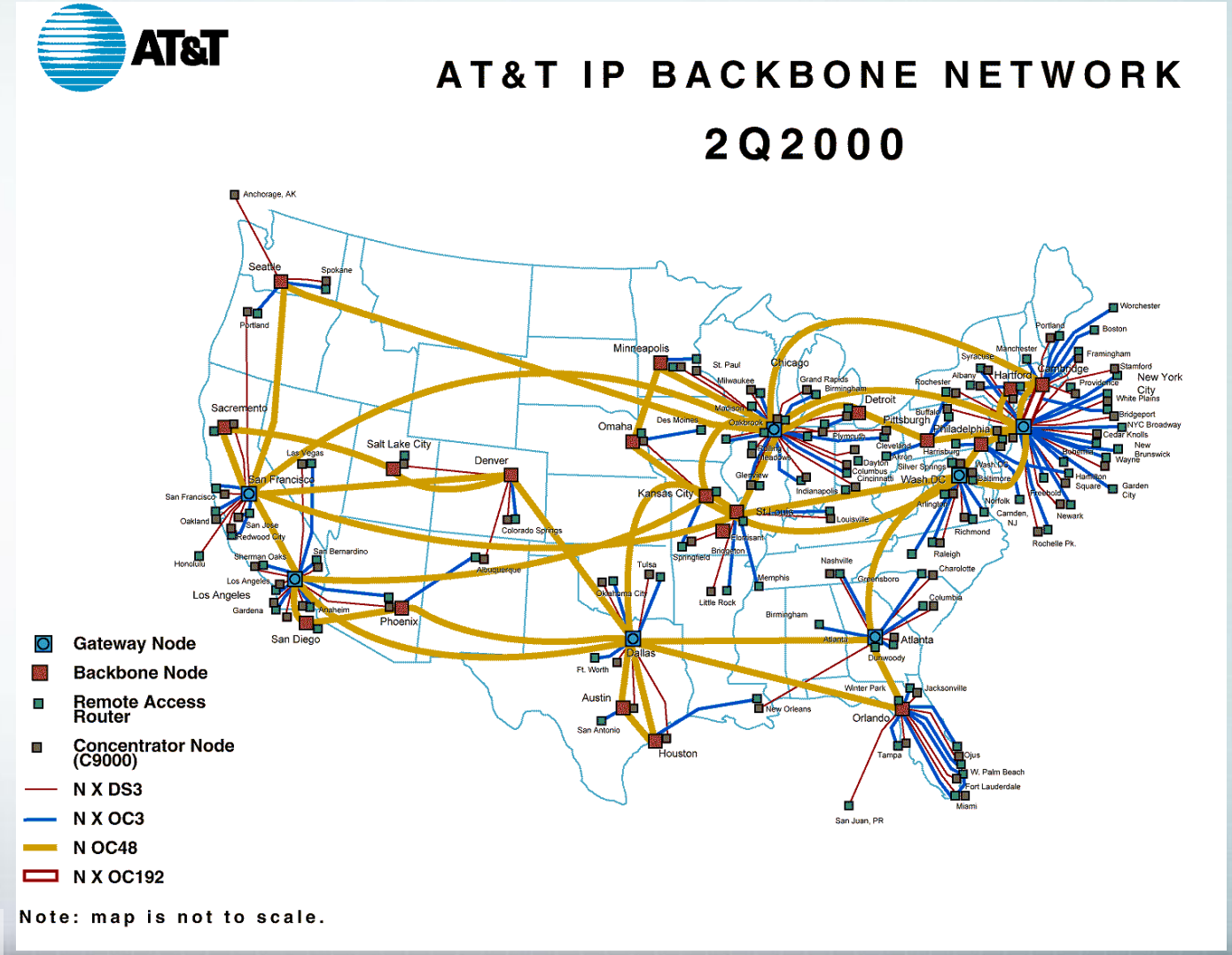
OPTIMIZACION

DISENO DE REDES

Parámetro: Numero de Nodos Reducido
Requisitos Similares



DISENO DE REDES



PROBLEMA

Proceso Simple

Requerimientos son Similares

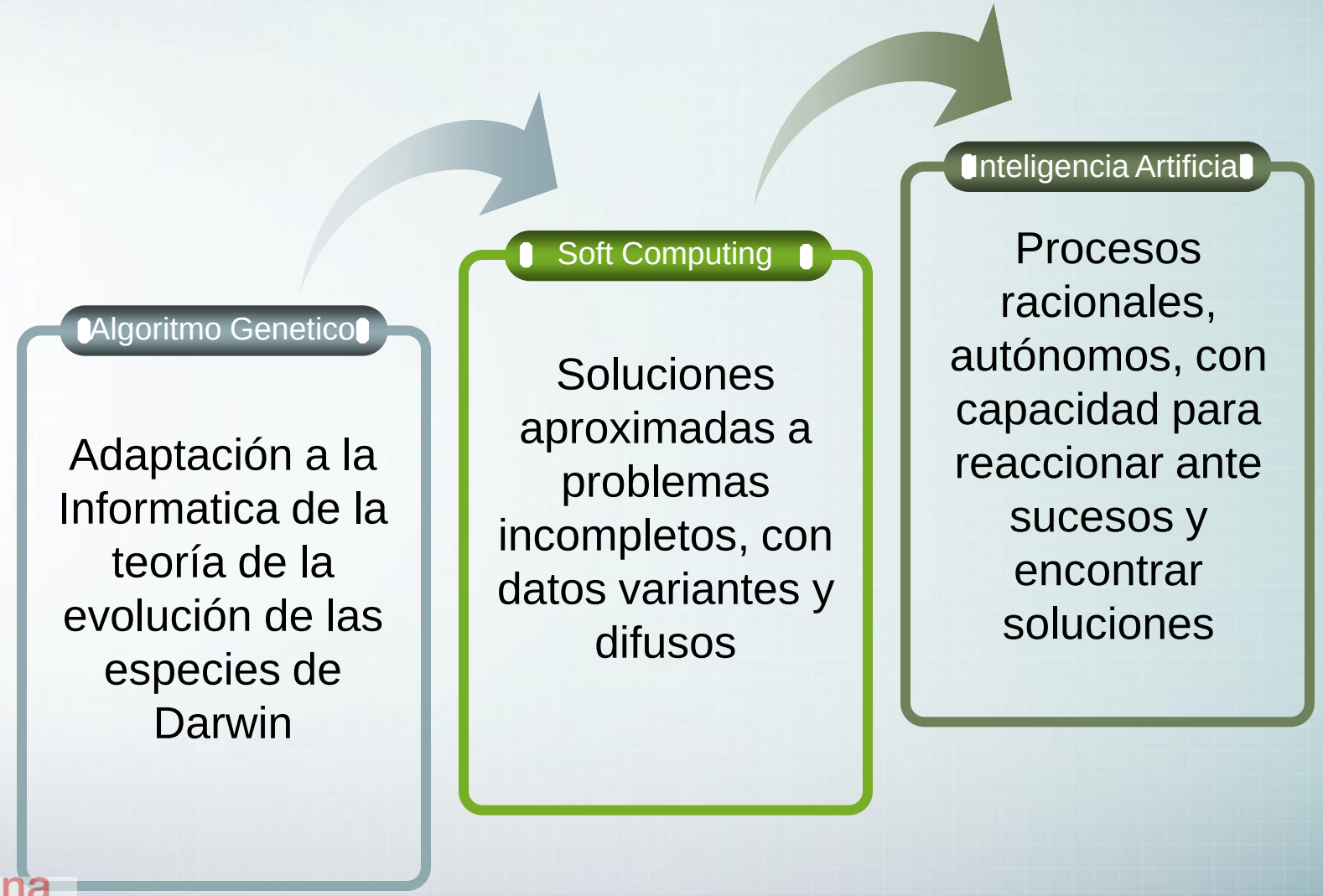
Numero de Nodos es Reducido

Proceso Complejo

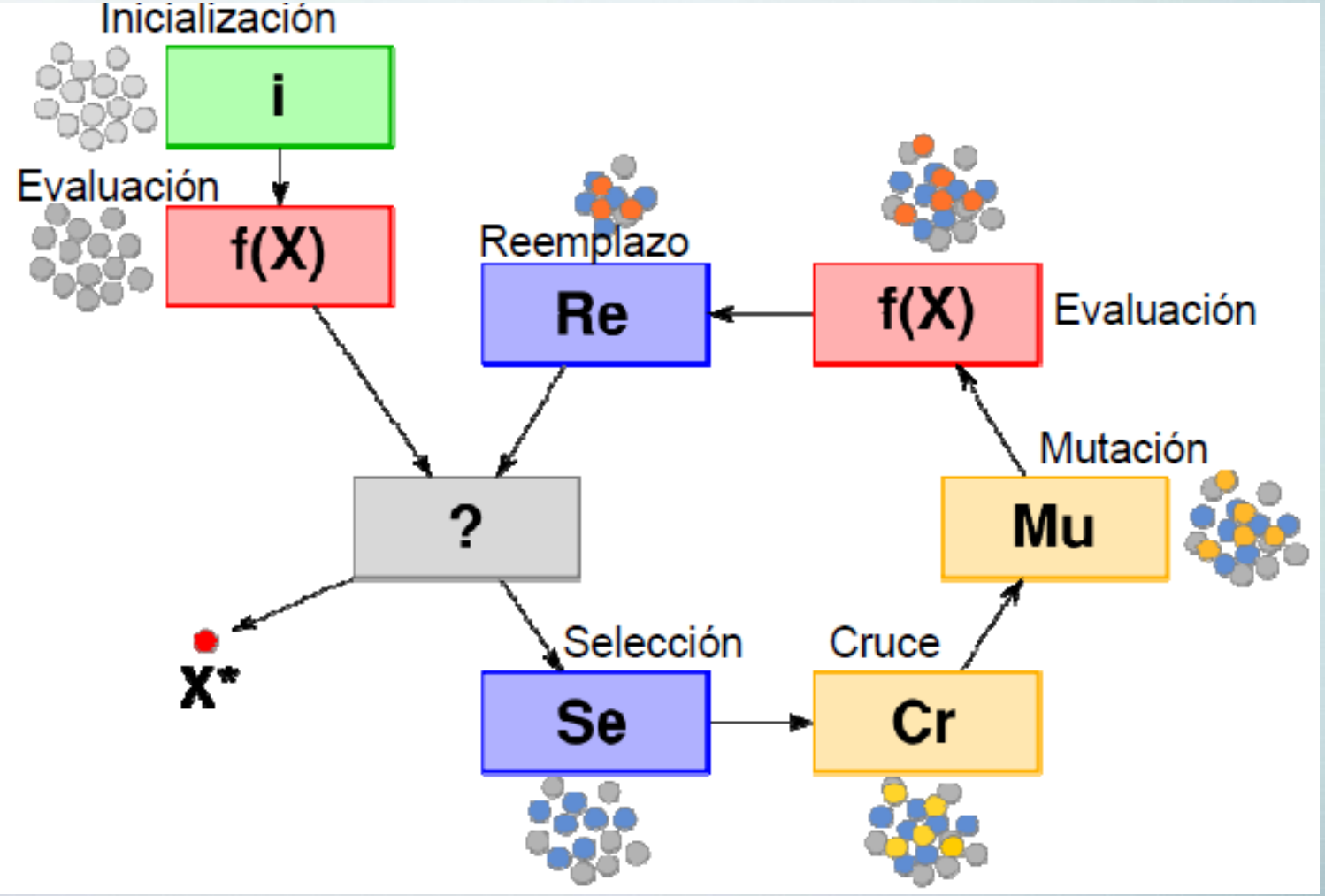
Requerimientos son Variados

Numero de Nodos es Amplio

SOLUCION



ESTRUCTURA DE UN ALGORITMO GENETICO



DETALLES DE LA SOLUCION

La solución del problema esta compuesta por dos algoritmos genéticos:

- algoritmo genético que forma grupos homogéneos de nodos, de manera que se puede determinar un núcleo backbone de la red.
- algoritmo genético que determine los enlaces adecuados para interconectar los nodos

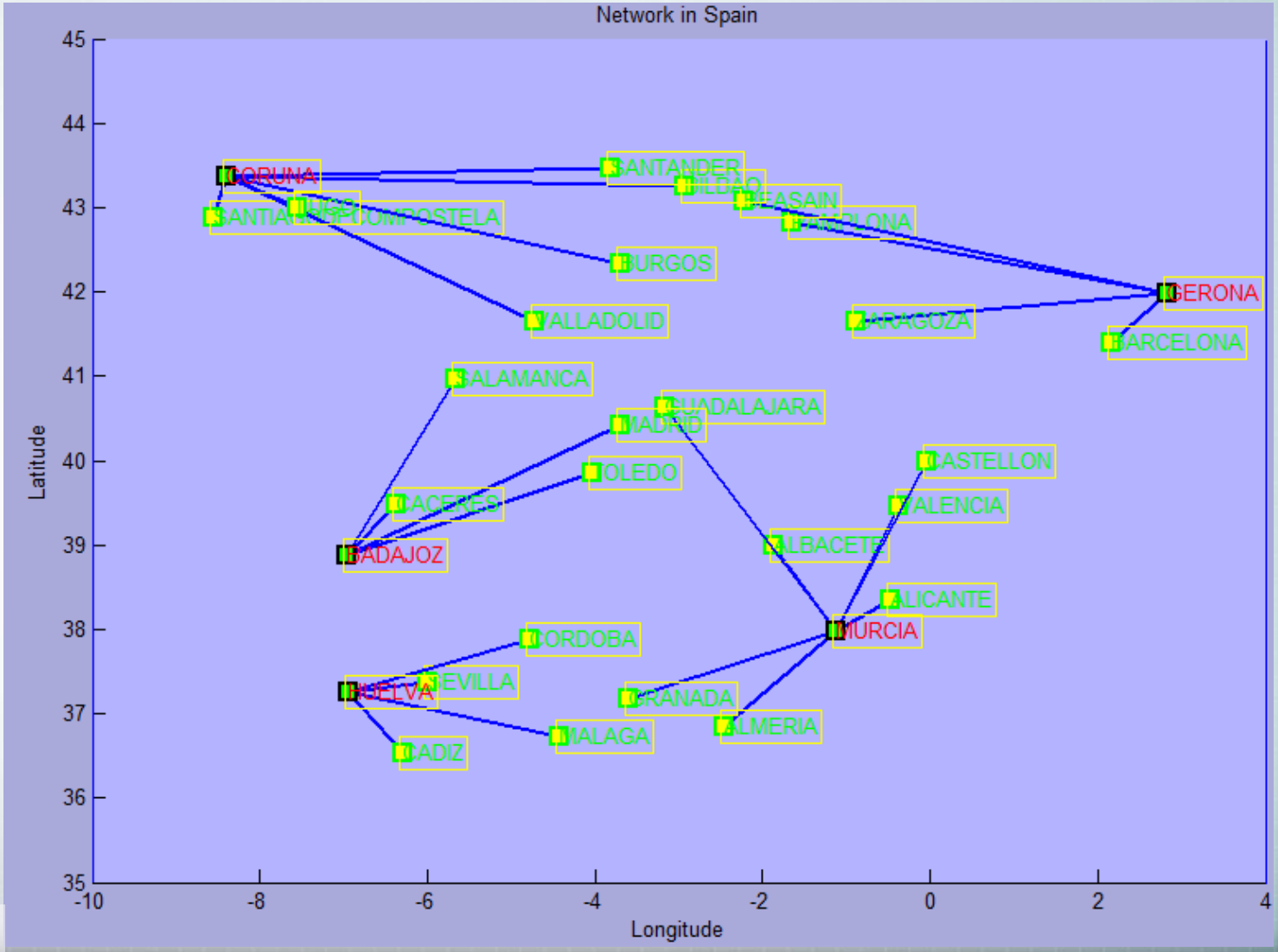
PARA EL PRIMER ALGORITMO GENETICO

FORMAR GRUPOS

Ecuación de la “gravedad de reilly” la cual se basa en la cantidad de población y distancia que existe entre dos puntos.

$$\mathbf{F}_{21} = -G\frac{m_1 m_2}{|\mathbf{r}_2 - \mathbf{r}_1|^2}\hat{\mathbf{u}}_{21}$$

PARA EL PRIMER ALGORITMO GENETICO



PARA EL PRIMER ALGORITMO GENETICO

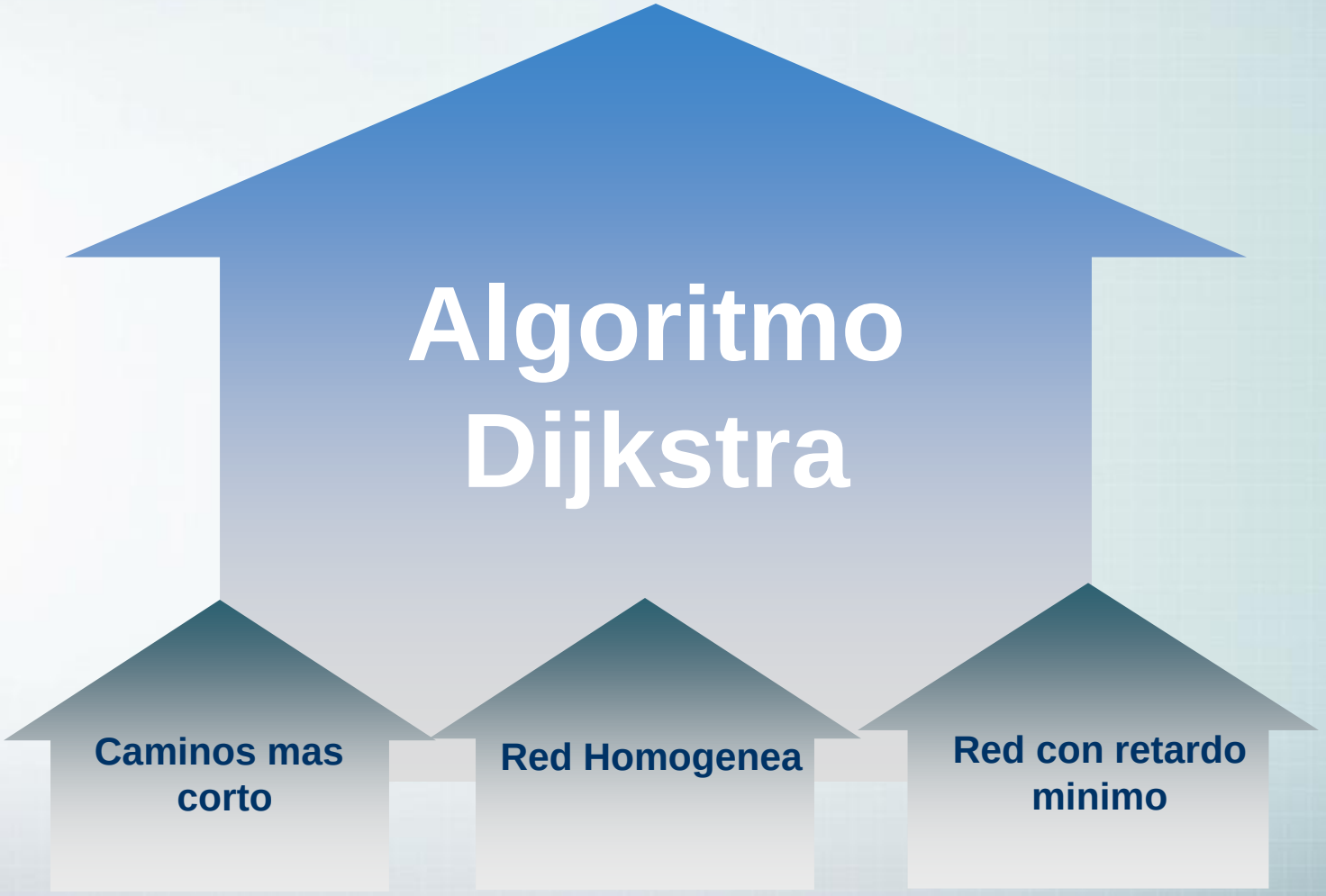
PARA EVALUAR LA HOMOGENEIDAD DE LOS GRUPOS FORMADOS POR LOS NODOS BACKBONE

Formula de la “Desviación Estándar”
si el valor devuelto es mínimo los grupos son similares

Expresión de la desviación estándar muestral:

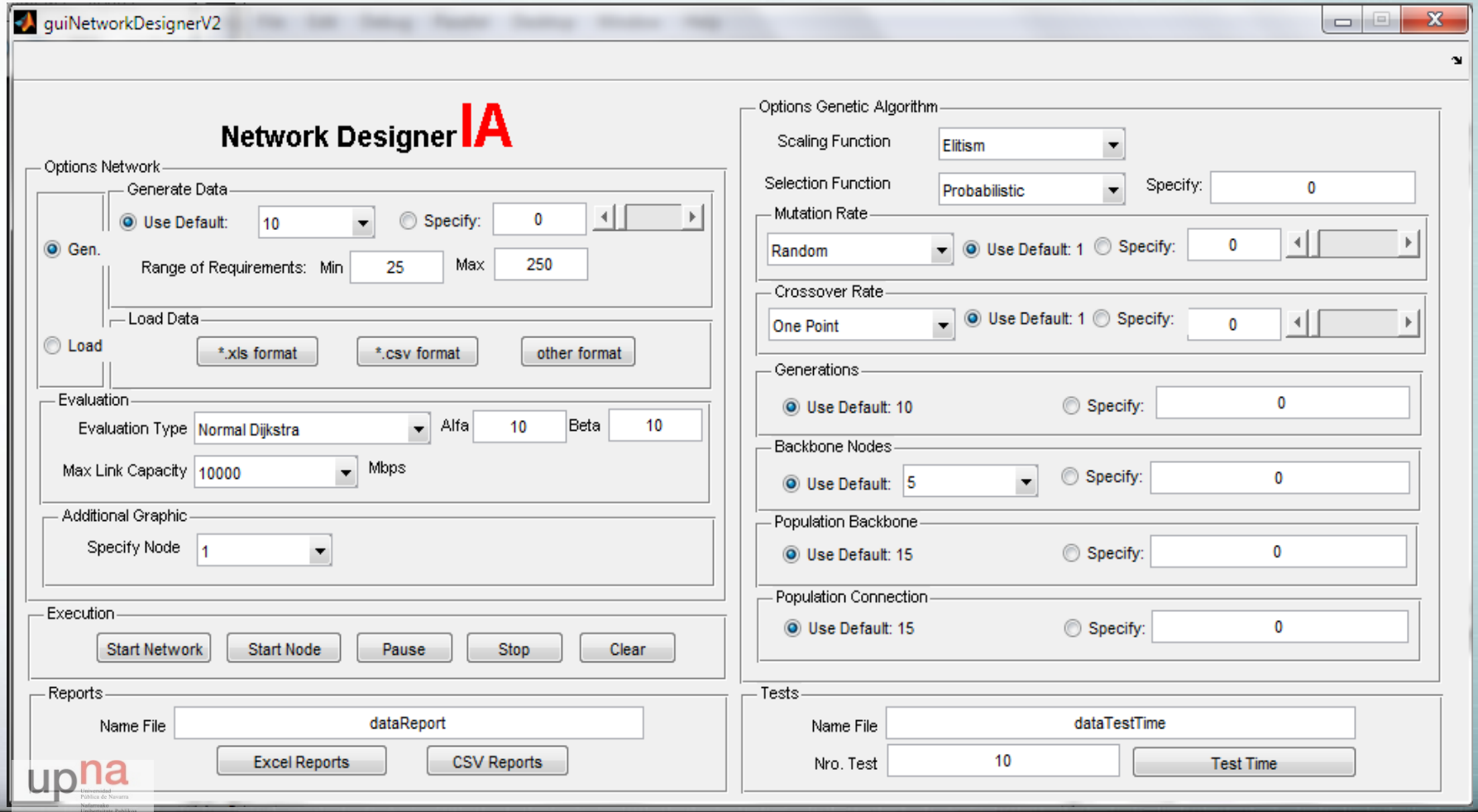
$$\sqrt{s^2} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

PARA EL SEGUNDO ALGORITMO GENETICO



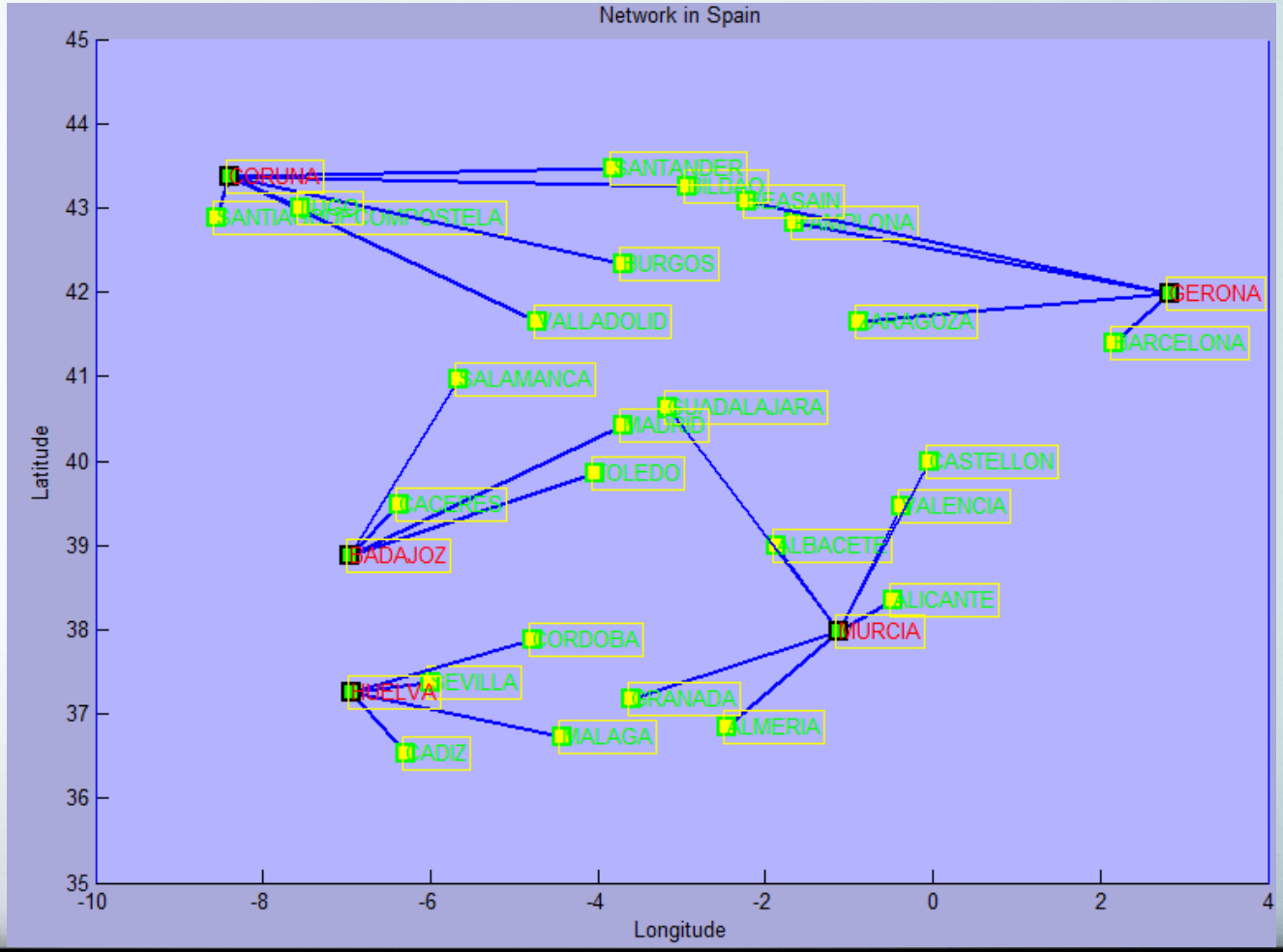
COMPLEMENTOS DEL SOFTWARE

- **Interfaz Grafica**



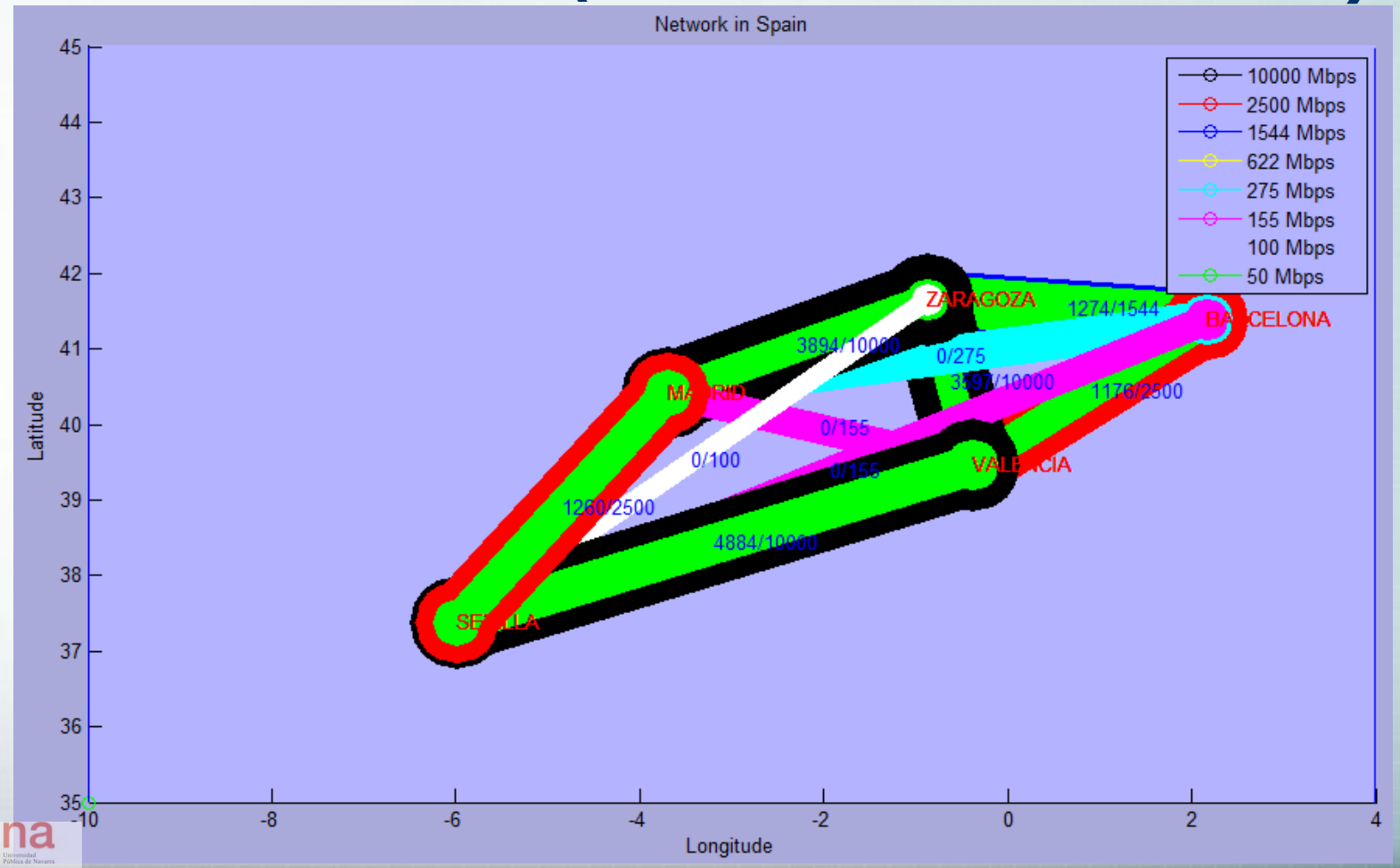
COMPLEMENTOS DEL SOFTWARE

- Graficar Solucion (Núcleo Backbone)



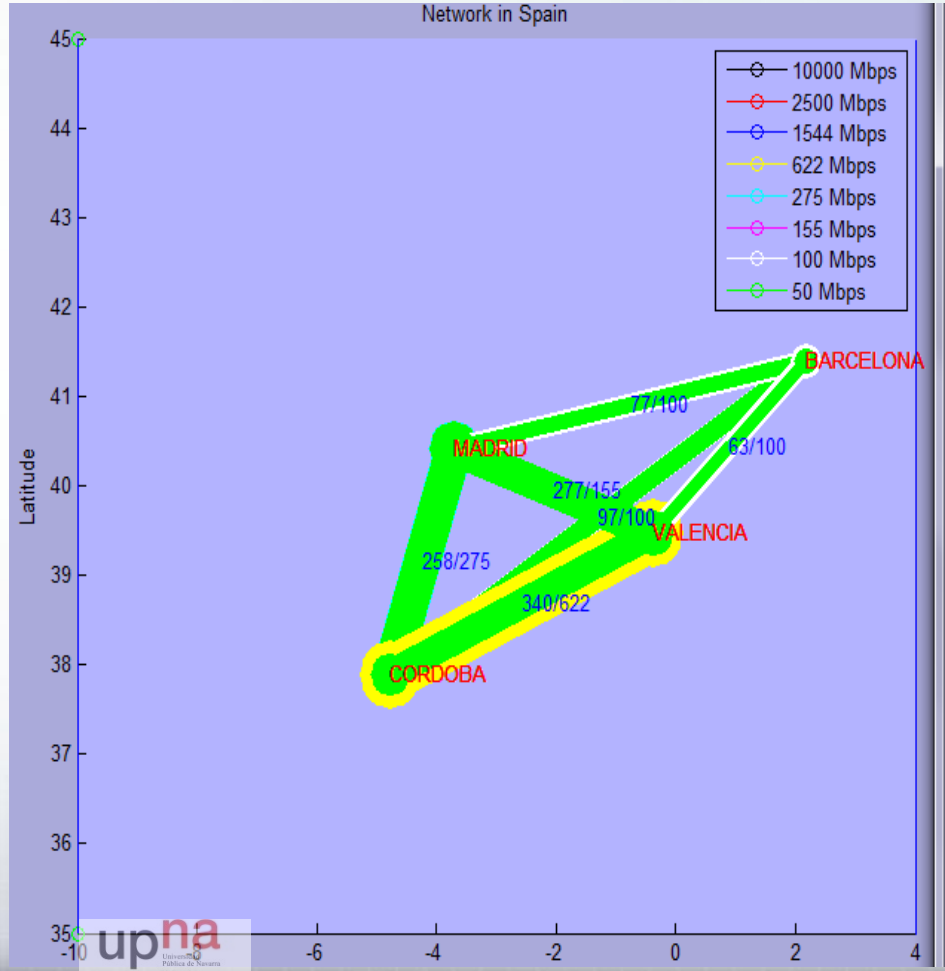
COMPLEMENTOS DEL SOFTWARE

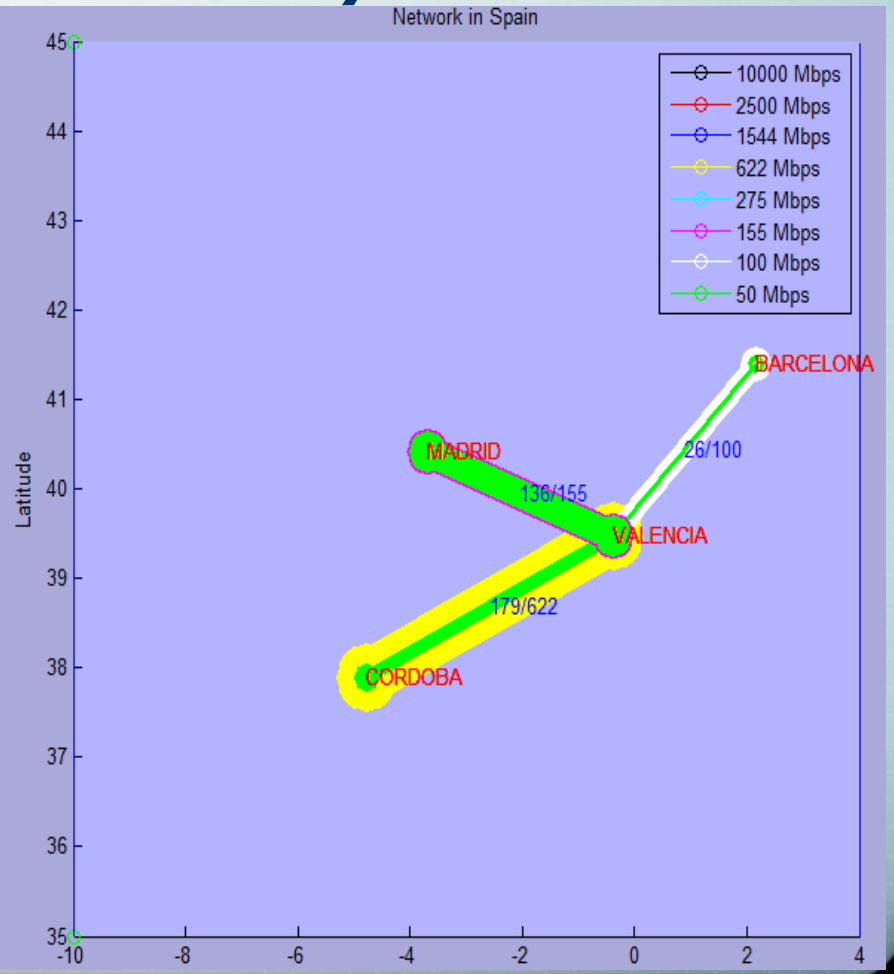
- Graficar Solucion (Interconexión de Nodos)



COMPLEMENTOS DEL SOFTWARE

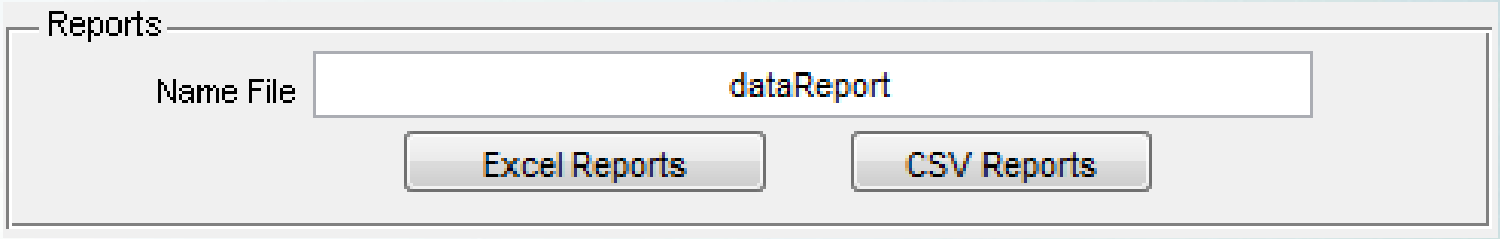
- Graficar Solucion (Grafico Parcial)





COMPLEMENTOS DEL SOFTWARE

Reportes



- 'requerimientos'
- 'coordenadas'
- 'nombreCiudades'
- 'poblacion'
- 'capacidadEnlaces'
- 'requerimientosFormados'
- 'vecNodosBackbone'
- 'poblacionFinal'
- 'vecRespuesta'
- 'tiempoEjecucion'

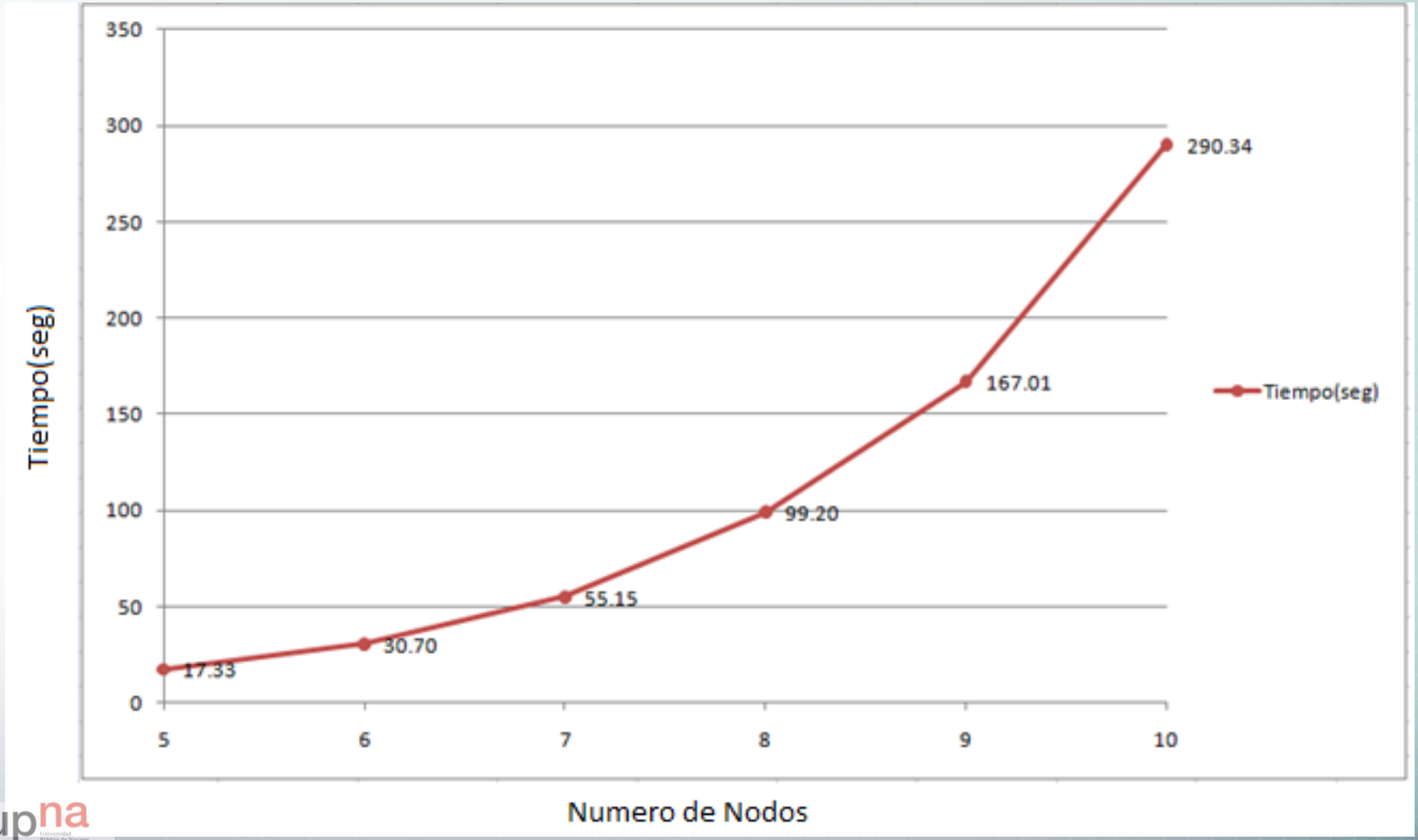
PRUEBAS

PRUEBA 1 :
Determina el tiempo de procesamiento que tarda el software en encontrar una solución, variando el número de nodos Backbone existentes en la red.

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Normal Dijkstra
- Mantener el Elitismo
- Selección probabilística
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- Población conformada por 15 cromosomas
- Numero de nodos Backbone (opción que se modifica)

PRUEBAS

RESULTADOS PRUEBA 1



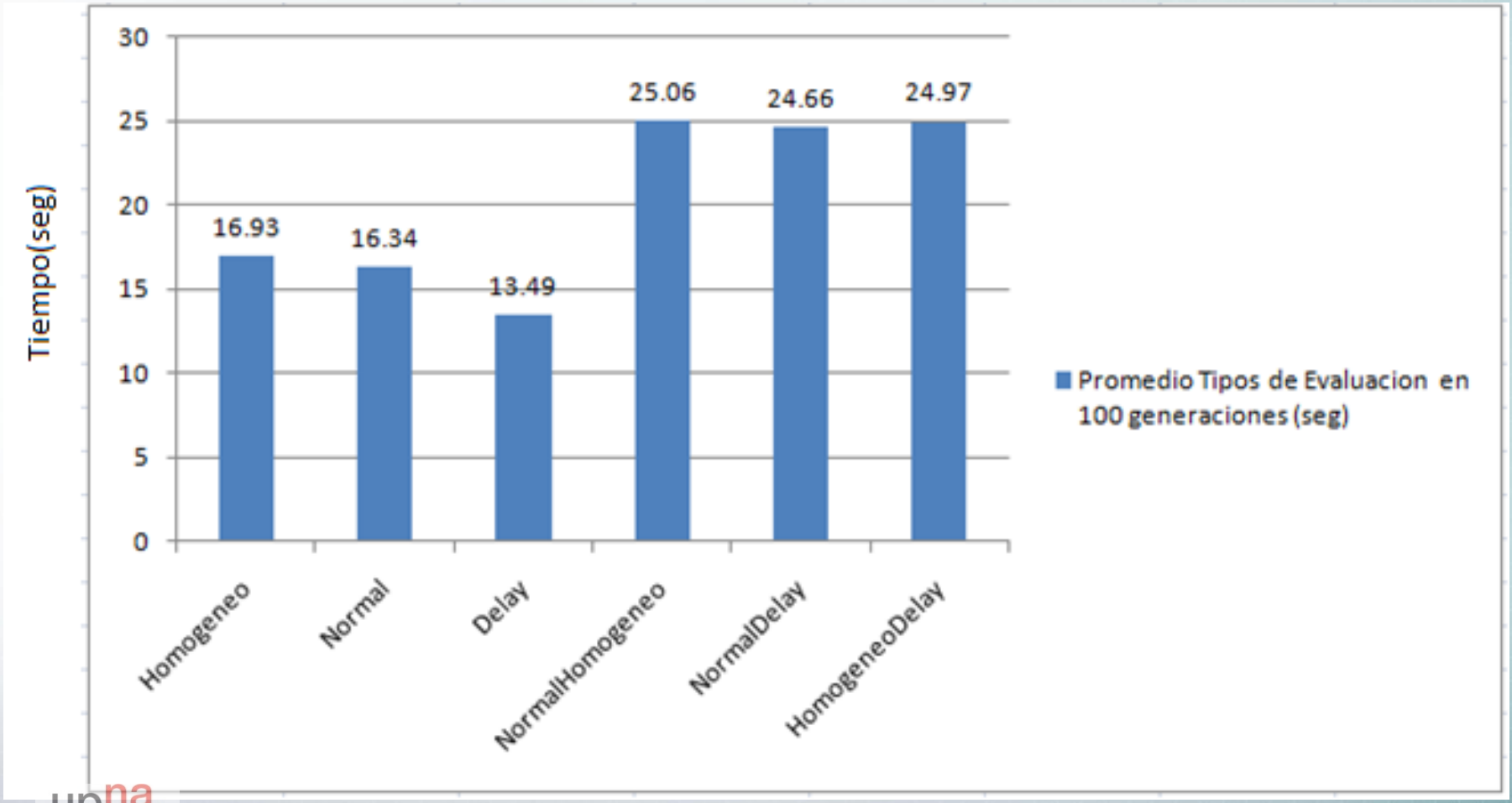
PRUEBAS

PRUEBA 2 :
Muestra el tiempo promedio de ejecución, existente entre los diferentes métodos de evaluación en el algoritmo genético.

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Mantener el Elitismo
- Selección probabilística
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Tipo de evaluación (opción que se modifica)

PRUEBAS

RESULTADOS PRUEBA 2



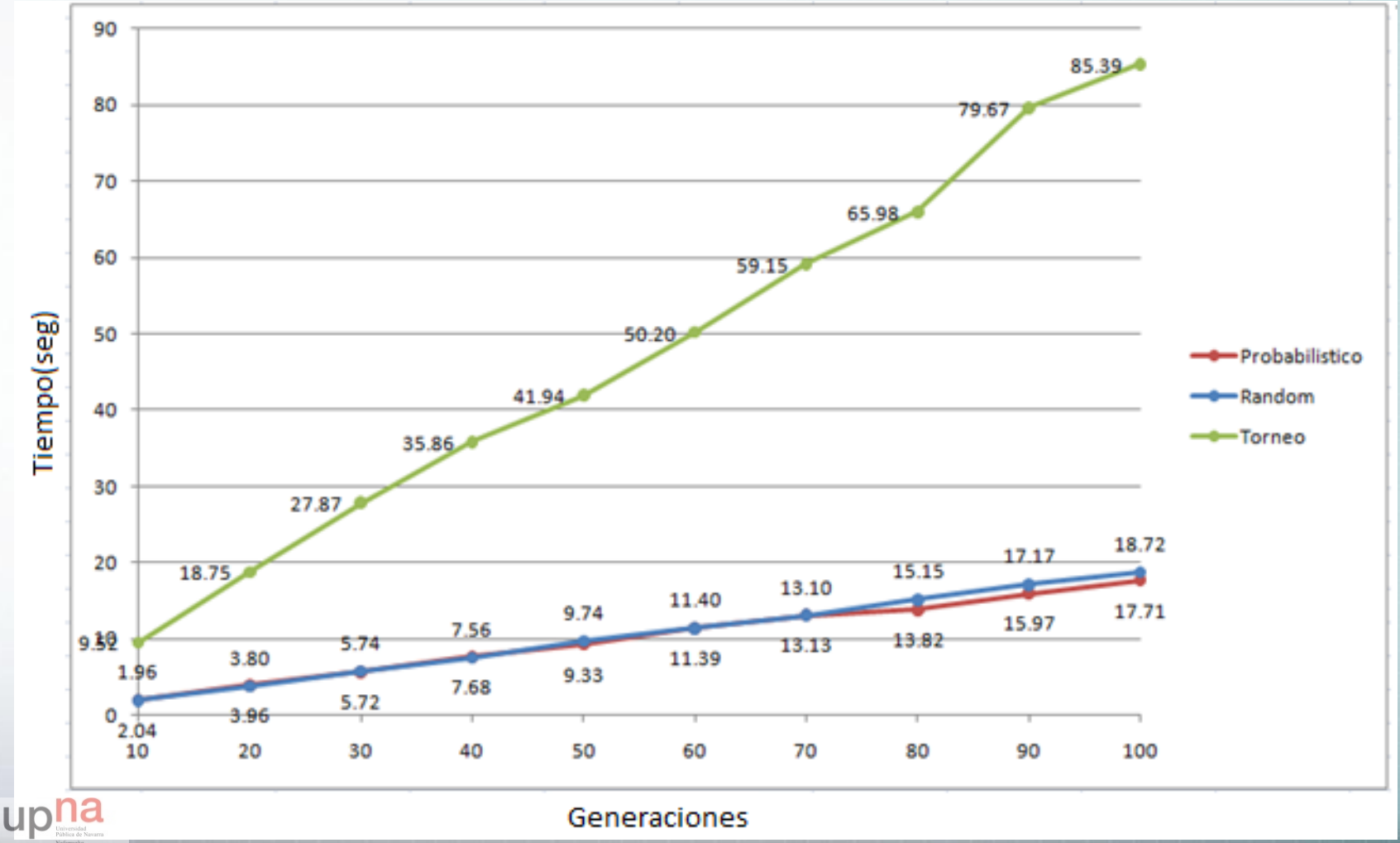
PRUEBAS

PRUEBA 3 :
Determina el tiempo de procesamiento del algoritmo genético, al hacer variar el tipo de selección utilizada para elegir a los cromosomas en una generación.

- Generación Aleatoriamente 30 nodos
- Requerimientos en el rango de 1 – 10 Mbps
- Evaluación Dijkstra Normal
- Mantener el Elitismo
- Si la selección es por torneo, el número de cromosomas por torneo es 5.
- Mutación ramdomica, con porcentaje de mutación del 1 porciento
- Cruce 1 punto, con porcentaje de cruce del 1 porciento
- Hasta 100 generaciones
- 5 nodos Backbone en la red
- Población conformada por 15 cromosomas
- Función de selección (opción que se modifica)

PRUEBAS

RESULTADOS PRUEBA 3



CONCLUSIONES

Con el desarrollo de los algoritmos geneticos en el PFC se puede observar las ventajas y desventajas que esta tecnica presenta

Ventajas:

- Multiples propuestas de soluciones
- Facilidad para representar el problema dentro del algoritmo genetico
- Exploracion del area de soluciones

Desventajas:

- Tiempo de procesamiento

TRABAJOS FUTUROS

•Integración de otros métodos de inteligencia artificial, por ejemplo los conjuntos difusos, para que el software tenga la capacidad de clasificar los sucesos que vayan ocurriendo en tiempo real y tome decisiones de una manera automática como la modificación del diseño de la red.

Gracias !!!

- Optimizacion del costo de enlaces en una red